

The Extreme-scale Scientific Software Stack (E4S) and its Promise for the Exascale Computing Era



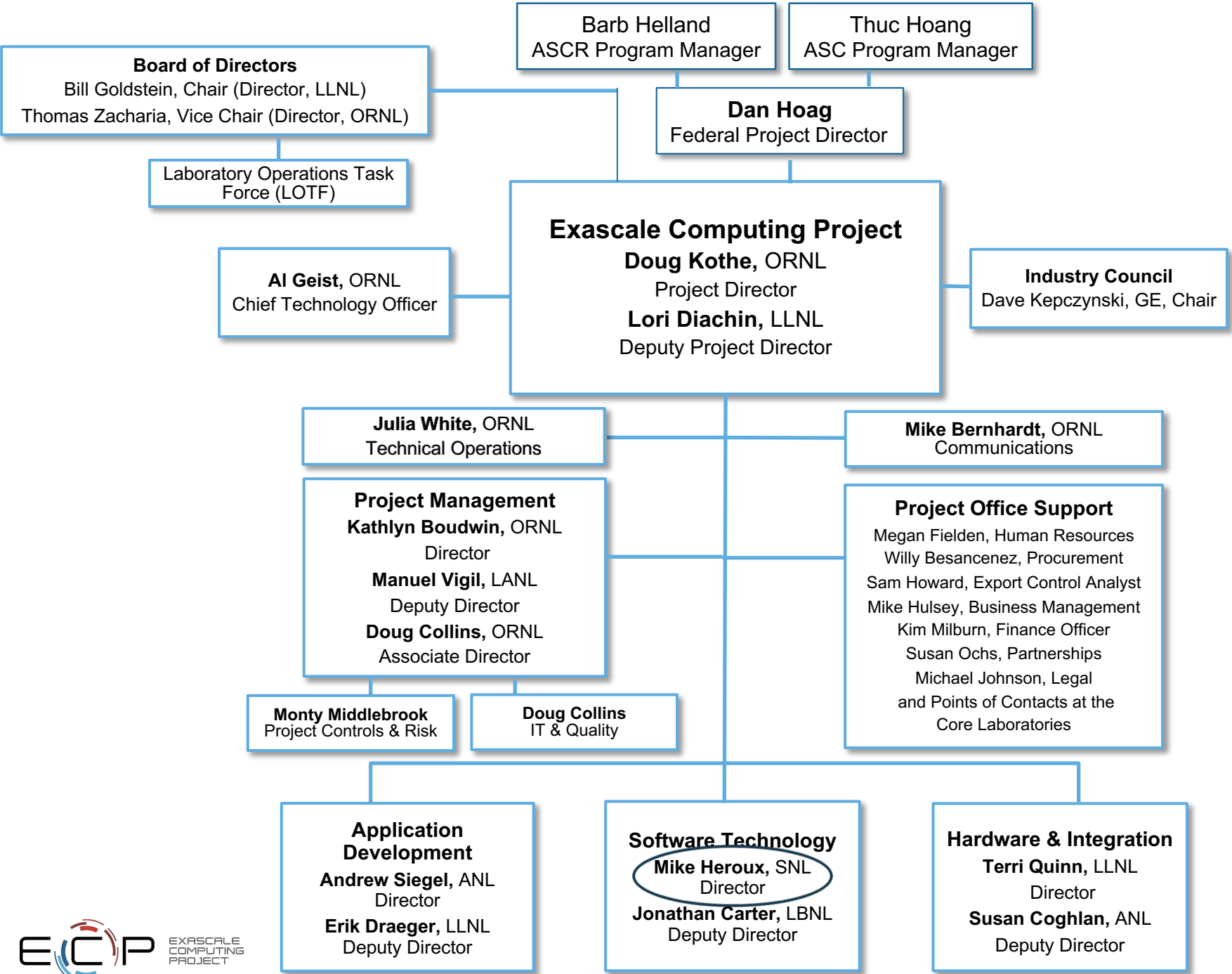
Michael A. Heroux, Sandia National Laboratories
Director of Software Technology, US Exascale Computing Project

ScalA19: 10th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems
November 18, 2019

ECP Context for E4S



ECP Organization

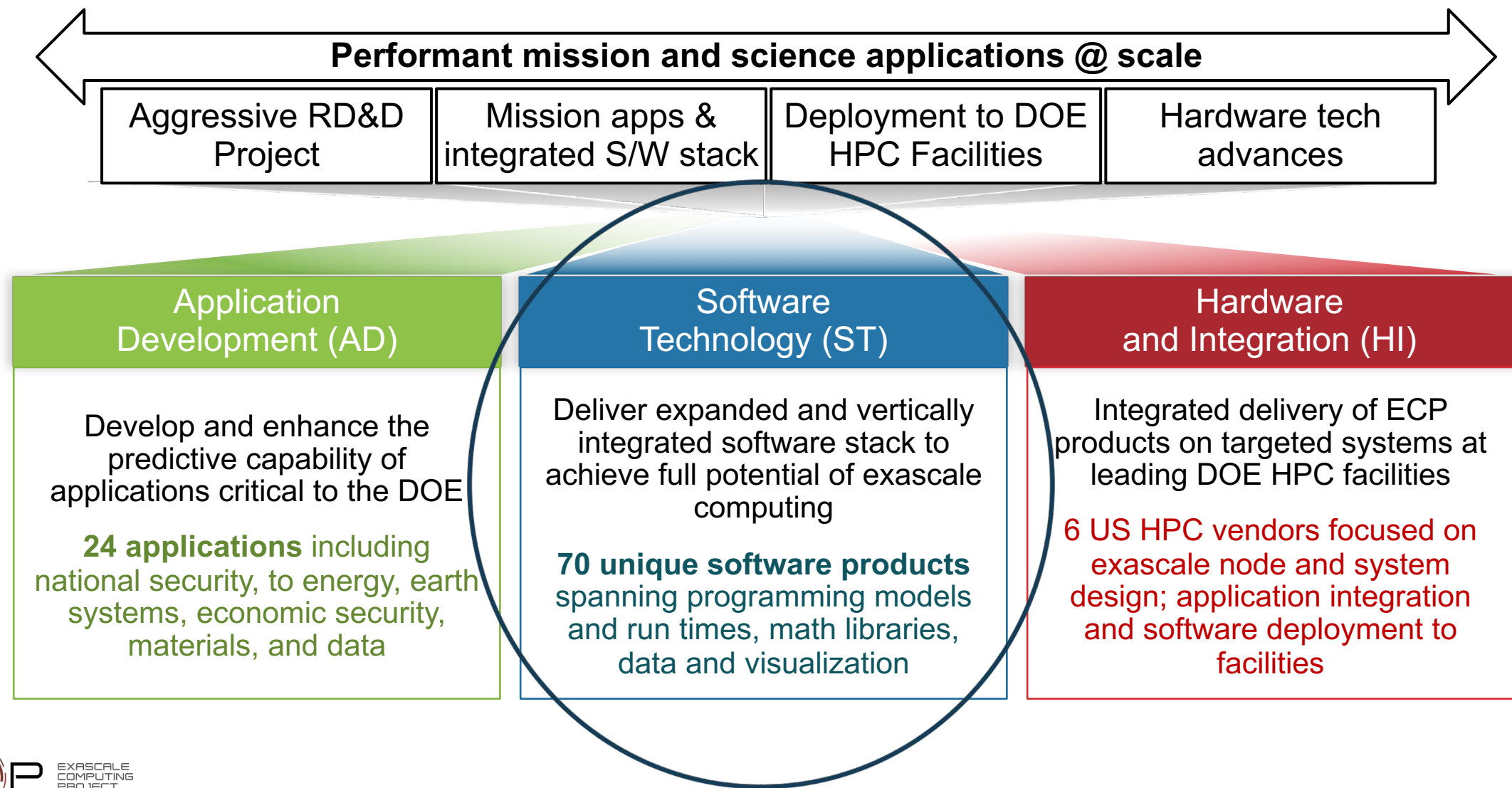


DOE HPC Facilities

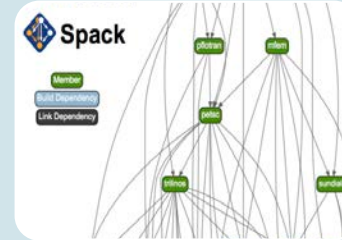
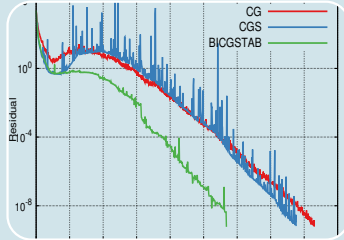
Core Laboratories



ECP's three technical areas have the necessary components to meet national goals



ECP software technologies overview



Programming Models & Runtimes

- Enhance and get ready for exascale the widely used MPI and OpenMP programming models (hybrid programming models, deep memory copies)
- Development of performance portability tools (e.g. Kokkos and Raja)
- Support alternate models for potential benefits and risk mitigation: PGAS (UPC++/GASNet), task-based models (Legion, PaRSEC)
- Libraries for deep memory hierarchy and power management

Development Tools

- Continued, multifaceted capabilities in portable, open-source LLVM compiler ecosystem to support expected ECP architectures, including support for F18
- Performance analysis tools that accommodate new architectures, programming models, e.g., PAPI, Tau

Math Libraries

- Linear algebra, iterative linear solvers, direct linear solvers, integrators and nonlinear solvers, optimization, FFTs, etc
- Performance on new node architectures; extreme strong scalability
- Advanced algorithms for multi-physics, multiscale simulation and outer-loop analysis
- Increasing quality, interoperability, complementarity of math libraries

Data and Visualization

- I/O via the HDF5 API
- Insightful, memory-efficient in-situ visualization and analysis – Data reduction via scientific data compression
- Checkpoint restart

Software Ecosystem

- Develop features in Spack necessary to support all ST products in E4S, and the AD projects that adopt it
- Development of Spack stacks for reproducible turnkey deployment of large collections of software
- Optimization and interoperability of containers on HPC systems
- Regular E4S releases of the ST software stack and SDKs with regular integration of new ST products

NNSA ST

- Open source NNSA Software projects
- Projects that have both mission role and open science role
- Major technical areas: New programming abstractions, math libraries, data and viz libraries
- Cover most ST technology areas
- Subject to the same planning, reporting and review processes

ECP Software Technology Leadership Team



Mike Heroux, [Software Technology](#) Director

Mike has been involved in scientific software R&D for 30 years. His first 10 were at Cray in the LIBSCI and scalable apps groups. At Sandia he started the Trilinos and Mantevo projects, is author of the HPCG benchmark for TOP500, and leads productivity and sustainability efforts for DOE.



Jonathan Carter, [Software Technology](#) Deputy Director

Jonathan has been involved in the support and development of HPC applications for chemistry, the procurement of HPC systems, and the evaluation of novel computing hardware for over 25 years. He currently a senior manager in Computing Sciences at Berkeley Lab.



Rajeev Thakur, [Programming Models and Runtimes](#) (2.3.1)

Rajeev is a senior computer scientist at ANL and most recently led the ECP Software Technology focus area. His research interests are in parallel programming models, runtime systems, communication libraries, and scalable parallel I/O. He has been involved in the development of open source software for large-scale HPC systems for over 20 years.



Jeff Vetter, [Development Tools](#) (2.3.2)

Jeff is a computer scientist at ORNL, where he leads the Future Technologies Group. He has been involved in research and development of architectures and software for emerging technologies, such as heterogeneous computing and nonvolatile memory, for HPC for over 15 years.



Lois Curfman McInnes, [Math Libraries](#) (2.3.3)

Lois is a senior computational scientist in the Mathematics and Computer Science Division of ANL. She has over 20 years of experience in high-performance numerical software, including development of PETSc and leadership of multi-institutional work toward sustainable scientific software ecosystems.



Jim Ahrens, [Data and Visualization](#) (2.3.4)

Jim is a senior research scientist at the Los Alamos National Laboratory (LANL) and an expert in data science at scale. He started and actively contributes to many open-source data science packages including ParaView and Cinema.



Todd Munson, [Software Ecosystem and Delivery](#) (2.3.5)

Todd is a computational scientist in the Math and Computer Science Division of ANL. He has nearly 20 years of experience in high-performance numerical software, including development of PETSc/TAO and project management leadership in the ECP CODAR project.



Rob Neely, [NNSA ST](#) (2.3.6)

Rob is an Associate Division Leader in the Center for Applied Scientific Computing (CASC) at LLNL, chair of the Weapons Simulation & Computing Research Council, and lead for the Sierra Center of Excellence. His efforts span applications, CS research, platforms, and vendor interactions.

We work on products applications need now and into the future

Key themes:

- Exploration/development of new algorithms/software for emerging HPC capabilities:
- High-concurrency node architectures and advanced memory & storage technologies.
- Enabling access and use via standard APIs.

Software categories:

- The next generation of well-known and widely used HPC products (e.g., MPICH, OpenMPI, PETSc)
- Some lesser used but known products that address key new requirements (e.g., Kokkos, RAJA, Spack)
- New products that enable exploration of emerging HPC requirements (e.g., SICM, zfp, UnifyCR)

Example Products	Engagement
MPI – Backbone of HPC apps	Explore/develop MPICH and OpenMPI new features & standards.
OpenMP/OpenACC –On-node parallelism	Explore/develop new features and standards.
Performance Portability Libraries	Lightweight APIs for compile-time polymorphisms.
LLVM/Vendor compilers	Injecting HPC features, testing/feedback to vendors.
Perf Tools - PAPI, TAU, HPCToolkit	Explore/develop new features.
Math Libraries: BLAS, sparse solvers, etc.	Scalable algorithms and software, critical enabling technologies.
IO: HDF5, MPI-IO, ADIOS	Standard and next-gen IO, leveraging non-volatile storage.
Viz/Data Analysis	ParaView-related product development, node concurrency.

ECP ST Subprojects

- WBS
- Name
- PIs
- Project Managers (PMs)

ECP ST Stats

- 33 L4 subprojects
- 10 PI/PC same
- 23 PI/PC different

WBS	WBS Name	CAM/PI	PC
2.3	Software Technology	Heroux, Mike, Carter, J.	-
2.3.1	Programming Models & Runtimes	Thakur, Rajeev	-
2.3.1.01	PMR SDK	Shende, Sameer	Shende, Sameer
2.3.1.07	Exascale MPI (MPICH)	Balaji, Pavan	Bayyapu, Neelima
2.3.1.08	Legion	McCormick, Pat	McCormick, Pat
2.3.1.09	PaRSEC	Bosilica, George	Carr, Earl
2.3.1.14	Pagoda: UPC++/GASNet for Lightweight Communication and Global Address Space Support	Baden, Scott	Hargrove, Paul (and PI)
2.3.1.16	SICM	Lang, Michael	Vigil, Brittney
2.3.1.17	OMPI-X	Bernholdt, David	TBD PMA Through ORNL PMO
2.3.1.18	RAJA/Kokkos	Trott, Christian Robert	Trott, Christian
2.3.1.19	Argo: Low-level resource management for the OS and runtime	Beckman, Pete	Gupta, Rinku
2.3.2	Development Tools	Vetter, Jeff	-
2.3.2.01	Development Tools Software Development Kit	Miller, Barton	Tim Haines
2.3.2.06	Exa-PAPI++: The Exascale Performance Application Programming Interface with Modern C++	Dongarra, Jack	Jagode, Heike
2.3.2.08	Extending HPCToolkit to Measure and Analyze Code Performance on Exascale Platforms	Mellor-Crummey, John	Mellor-Crummey, John
2.3.2.10	PROTEAS-TUNE	Vetter, Jeff	Glassbrook, Dick
2.3.2.11	SOLLVE: Scaling OpenMP with LLVM for Exascale	Chapman, Barbara	Kong, Martin
2.3.2.12	FLANG	McCormick, Pat	Perry-Holby, Alexis
2.3.3	Mathematical Libraries	McInnes, Lois	-
2.3.3.01	Extreme-scale Scientific xSDK for ECP	Yang, Ulrike	Yang, Ulrike
2.3.3.06	Preparing PETSc/TAO for Exascale	Smith, Barry	Munson, Todd
2.3.3.07	STRUMPACK/SuperLU/FFTX: sparse direct solvers, preconditioners, and FFT libraries	Li, Xiaoye	Li, Xiaoye
2.3.3.12	Enabling Time Integrators for Exascale Through SUNDIALS/ Hypre	Woodward, Carol	Woodward, Carol
2.3.3.13	CLOVER: Computational Libraries Optimized Via Exascale Research	Dongarra, Jack	Carr, Earl
2.3.3.14	ALExa: Accelerated Libraries for Exascale/ForTrilinos	Turner, John	TBD PMA Through ORNL PMO
2.3.4	Data and Visualization	Ahrens, James	-
2.3.4.01	Data and Visualization Software Development Kit	Atkins, Chuck	Atkins, Chuck
2.3.4.09	ADIOS Framework for Scientific Data on Exascale Systems	Klasky, Scott	TBD PMA Through ORNL PMO
2.3.4.10	DataLib: Data Libraries and Services Enabling Exascale Science	Ross, Rob	Ross, Rob
2.3.4.13	ECP/VTK-m	Moreland, Kenneth	Moreland, Kenneth
2.3.4.14	VeloC: Very Low Overhead Transparent Multilevel Checkpoint/Restart/Sz	Cappello, Franck	Ehling, Scott
2.3.4.15	ExaIO - Delivering Efficient Parallel I/O on Exascale Computing Systems with HDF5 and Unify	Byna, Suren	Bagha, Neelam
2.3.4.16	ALPINE: Algorithms and Infrastructure for In Situ Visualization and Analysis/ZFP	Ahrens, James	Turton, Terry
2.3.5	Software Ecosystem and Delivery	Munson, Todd	-
2.3.5.01	Software Ecosystem and Delivery Software Development Kit	Willenbring, James M	Willenbring, James M
2.3.5.09	SW Packaging Technologies	Gamblin, Todd	Gamblin, Todd
2.3.6	NNSA ST	Neely, Rob	-
2.3.6.01	LANL ATDM	Mike Lang	Vandenbusch, Tanya Marie
2.3.6.02	LLNL ATDM	Becky Springmeyer	Gamblin, Todd
2.3.6.03	SNL ATDM	Jim Stewart	Trujillo, Gabrielle

Iterative, incremental
delivery of a modular,
interoperable, and
deployable software
stack



Delivering a Modular, Interoperable, and Deployable Software Stack

Objectives

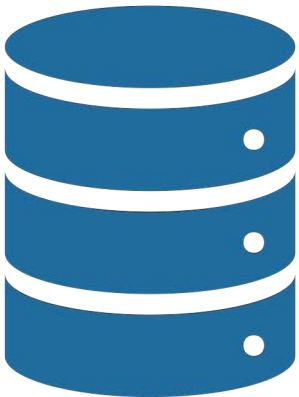
- Lower barrier to use ST products
- Lower barrier to enable Facilities to install all or parts
- Enable interoperability between ST products
- Enable uniform APIs where possible
- Establish an open, hierarchical software architecture to enable collaboration with other US Agencies and international HPC software institutions.

Challenges

- Large diverse group of ST products
- Different project management styles
- Lack of initial drivers for integration
- Complex software ecosystem with many players.

ECP ST Planning Process: Hierarchical, three-phase, cyclical

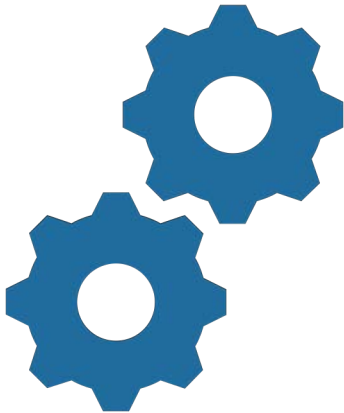
Baseline



FY20–23 Baseline Plan High level Definitions

- Q2 FY19 start
- FY20 Base plan
- FY21–23 planning packages

Annual Refinement



FY Refine Baseline Plan As Needed Basic activity definitions

- 6 months prior to FY
- 4–6 P6 Activities/year
- Each activity:
 - % annual budget
 - Baseline start/end
 - High level description

Per Activity



Detailed Plan Complete activity definitions

- 8 weeks prior to start
- High-fidelity description
- Execution strategy
- Completion criteria
- Personnel details

Two-level Change Control	
Changes to Cost, Scope, and Schedule	
Minor	Major
Lightweight Review in Jira, L3 and L2 leads	Change Control Board Review, ECP leadership
Variance Recorded in Jira	
Proceed with Execution	

KPP-3 Definition

KPP ID	Description of Scope	Threshold KPP	Objective KPP	Verification Action/Evidence
KPP-1	Performance of scientific and national security applications relative to today's performance	50% of selected applications achieve Figure of merit* improvement ≥ 50	100% of selected applications achieve Figure of merit improvement stretch goal	Independent assessment of measured results and report that threshold goal is met
KPP-2	Broaden the reach of exascale science and mission capability	50% of selected applications can execute their challenge problem*	100% of selected applications can execute their challenge problem stretch goal	Independent assessment of mission application readiness
KPP-3	Productive and Sustainable Software Ecosystem	Software teams meet 50% of their weighted impact goals*	Software teams meet 100% of their weighted impact stretch goals	Independent assessment verifying threshold goal is met
KPP-4	Enrich the HPC Hardware Ecosystem	Vendors meet 80% of all the PathForward milestones	Vendors meet 100% of all the PathForward milestones	Independent assessment of the impact and timeliness of PathForward milestones

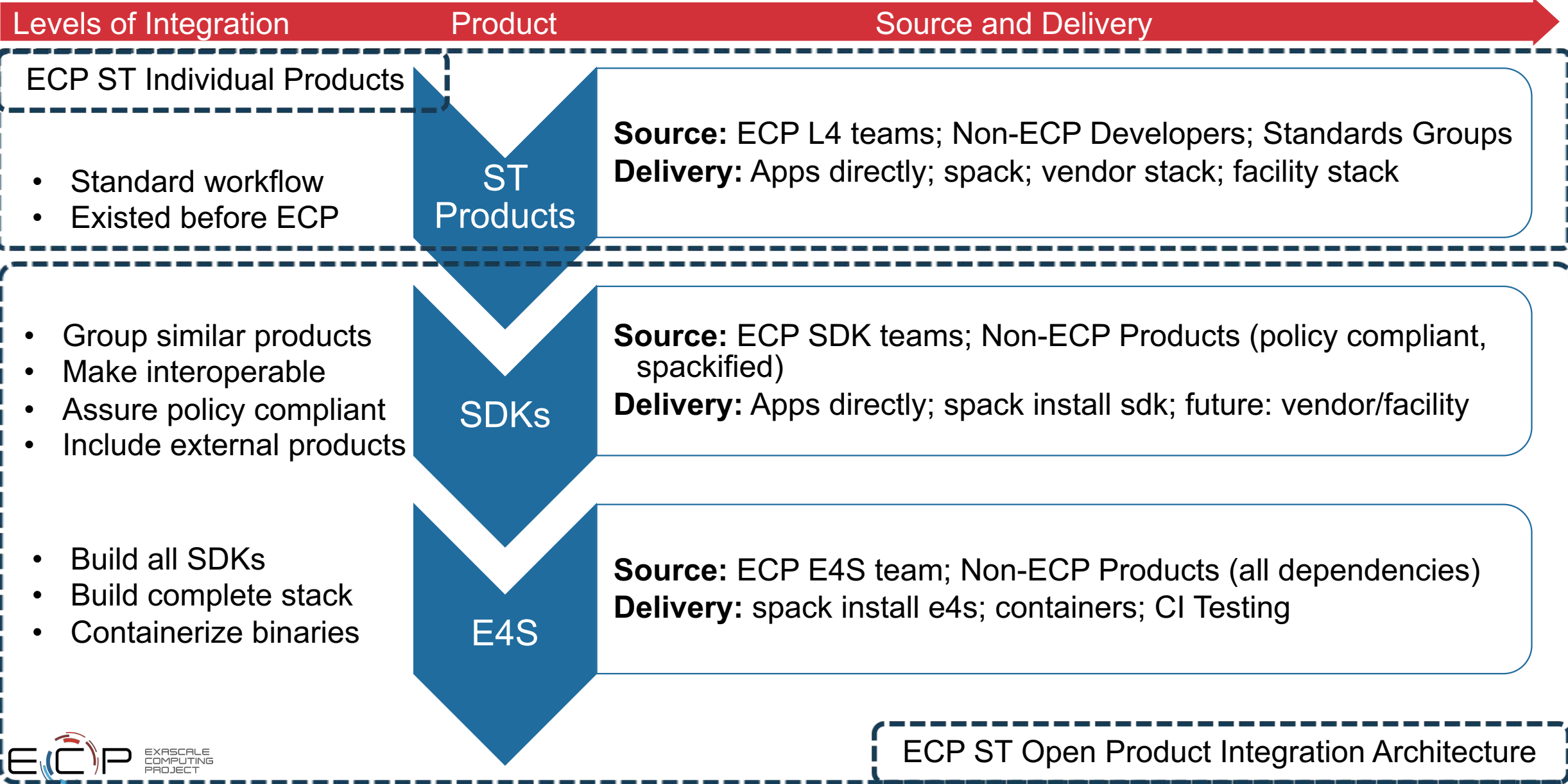
KPP-3: Focus on capability integration

- **Capability:** Any significant product functionality, including existing features adapted to the pre-exascale and exascale environments, that can be integrated into a client environment.
- **Capability Integration:** Complete, sustainable integration of a significant product capability into a client environment in a pre-exascale environment (tentative score) and in an exascale environment (confirmed score).
- **Product:** Capabilities are integrated into primary products chosen from a product dictionary.
- **Primary products examples:**
 - MPI is commonly known by users. MPICH and OpenMPI both provide implementations of that product.
 - Fortran is a product. Flang is a particular Fortran product. LLVM is a backend for some Fortran compilers.
 - FFT is a product. FFTX, FFT-ECP provide FFT capabilities through interchangeable interfaces.
 - C++ is a product. Clacc provides capabilities for Clang, as does LLVM.
- **Details:** See Confluence page export.

E4S Overview



Software Technology Ecosystem



Extreme-scale Scientific Software Stack (E4S)

- E4S: A Spack-based distribution of ECP ST and related and dependent software tested for interoperability and portability to multiple architectures
 - Provides distinction between SDK usability / general quality / community and deployment / testing goals
 - Will leverage and enhance SDK interoperability thrust
-
- Oct 2018: E4S 0.1 - 24 full, 24 partial release products
 - Jan 2019: E4S 0.2 - 37 full, 10 partial release products
 - Nov 2019: E4S 1.0 - 50 full, 5 partial release products



e4s.io

Lead: Sameer Shende
(U Oregon)

E4S 1.0 Full Release: 50 ECP Packages and all dependencies

- Adios
- AML
- Argobots
- Bolt
- Caliper
- Darshan
- Dyninst
- Faodel
- Flecsi
- Gasnet
- GEOPM
- Gotcha
- HDF5
- HPCToolkit
- Hydre
- Kokkos
- Legion
- Libnm
- Libquo
- Magma
- Mercury
- MFEM
- MPICH
- MPIFileUtils
- Ninja
- OpenMPI
- PAPI
- Papyrus
- Parallel netCDF
- PDT
- PETSc
- Qthreads
- Raja
- Rempi
- SCR
- Spack
- Strumpack
- Sundials
- SuperLU
- SZ
- Tasmanian
- TAU
- Trilinos
- Turbine

[illegible]

Packages installed using Spack

- Umpire
- UnifyFS
- UPC++ Veloc
- Zfp

**All ST products
will be released
through E4S**

ECP ST Product Dictionary

Created by Mike Heroux, last modified by Vivek Kale on 2019-10-05

Official Product Dictionary

- Standardizes Product Definitions
- Enables consistent AD-CD-ST dependencies

The ECP Software Technology (ST) Product Dictionary is the official list of publicly recognized names to which ECP ST efforts contribute. While ST teams use an expanded product namespace, the list on this page indicates the eventual access point for ST product development efforts.

This table lists in **bold** only those products that are typically recognizable to users. We call these **primary products**. Examples:

1. MPI is commonly known by users. MPICH and OpenMPI both provide implementations of that product.
2. Fortran is a product. Flang is a particular Fortran product. LLVM is a backend for some Fortran compilers.
3. FFT is a product. FFTX, FFT-ECP provide FFT capabilities through interchangeable interfaces.
4. C++ is a product. Clacc provides capabilities for Clang, as does LLVM.

Subproducts are listed underneath with the primary product name as a prefix.

In some cases, a product may be listed as a primary product even if it is not widely recognized by the user community. In this case, the product developer needs to be able to address these additional criteria:

1. The product is intended for stand-alone delivery and not be included in some other product in the future.
2. There is a credible sustainability path for the product and the development team provides some evidence that it can support the product in a sustainable way, including staffing and funding.

Deployment scope is meant to give others a sense of how widely integrated a product is in the HPC ecosystem and how far along it is in maturing toward usability. The deployment scope categories are:

- **Broad:** Widely used in the HPC ecosystem, expected to be available and usable by many applications.
- **Moderate:** Some use in applications but not ubiquitous.
- **Experimental:** Still under development and used by collaborators and friendly users.

Product	URL	Description/Notes	Deployment Scope	Technical Area	Point of Contact
1. ADIOS	https://github.com/ornladios/ADIOS2	I/O and data management library for storage I/O, in-	Broad	Data & Viz	@Scott Klasky

2. AID	
AID: STAT	https://github.com/LLN
AID: Archer	https://github.com/PRL
AID: FLIT	https://github.com/PRL

Product	URL	Description/Notes	Deployment Scope	Technical Area	Point of Contact
		communication in a PGAS model.			
66. Vendor Stack		ECP ST design, development and demonstration efforts that are integrated into one or more of the Vendor software stacks	Experimental	Software Ecosystem	TBD
67. VeloC	https://github.com/ECP-VeloC/VELOC	Scalable checkpoint-restart library.	Broad	Data & Viz	@Franck Cappello
68. VTK-m	http://m.vtk.org	Parallel on-node visualization toolkit.	Broad	Data & Viz	@Kenneth Moreland
69. xSDK	https://xsdk.info	Math libraries meta product combining the most popular HPC math libraries into a compatible collection.	Moderate	Math libraries	@Ulrike Meier Yang
70. zfp	https://github.com/LLNL/zfp	In-memory data compression library.	Moderate	Data & Viz	@Peter Lindstrom @Terry Turton

70 Official Products

- Will change over time
- Establishes delivery plan

Partial List of Jira Dependency Issues

Search

Save as

Share

Export

Tools

Project: All

Dependency

Status: All

Assignee: All

Contains text

More

Search

Advanced

1-50 of 814

Columns

T	Key	Summary	Assignee	Reporter	Status ↑	Resolution	Created	Due	Baseline end date	Links	Development
	INT-691	PETSc/TAO ↔ Spack	Unassigned	Todd Munson	APPROVED	Approved	2019-09-10				
	INT-686	HDF5 ↔ PETSc/TAO	Unassigned	Todd Munson	APPROVED	Approved	2019-09-10				
	INT-690	PETSc/TAO ↔ xSDK	Unassigned	Todd Munson	APPROVED	Approved	2019-09-10				
	INT-687	MPI-IO ↔ PETSc/TAO	Unassigned	Todd Munson	APPROVED	Approved	2019-09-10				
	INT-689	OpenMP ↔ PETSc/TAO	Unassigned	Todd Munson	APPROVED	Approved	2019-09-10				
	INT-677	libEnsemble ↔ xSDK	Unassigned	Todd Munson	APPROVED	Approved	2019-09-10				
	INT-681	BLAS ↔ PETSc/TAO	Unassigned	Todd Munson	APPROVED	Approved	2019-09-10				

Jira Dependency Issue Type

- Official ST and AD Product lists enable rigorous dependency management.
- New Jira Dependency issue type.
- Explicit dependency connections between AD/CD/ST products

Edit Issue : INT-1065

Configure Fields ▾

Reporter* ECP Support

Start typing to get a list of possible matches.

6

Producer*

Programming Models and Runtimes ▾

Legion ▾

Select the application code or product name that the Consumer *depends on*.

7

Consumer*

Data Analytics and Optimization ▾

CANDLE ▾

Select the application code or ST product that *depends on* the Producer.

8

Dependency Level*

☒ Critical

☐ Important

☐ Interested

Critical: The team is entirely dependent on the producer for this functionality, and there are no alternatives available.

Important: The team believes this producer is the best source for this functionality, but alternative sources exist.

Interested: The team is interested enough in the functionality that they are likely to try to adopt it into their work.

9

Functionality Description

ENTER BRIEF DESCRIPTION HERE

Enter a brief description of the functionality that this Producer provides this Consumer.

10

Trigger Event*

Unknown ▾

Provide the event/quarter you expect this dependency will be needed by.

Optional Items

11

Linked Issues

blocks ▾

Issue

Begin typing to search for issues to link. If you leave it blank, no link will be made.

Attachment

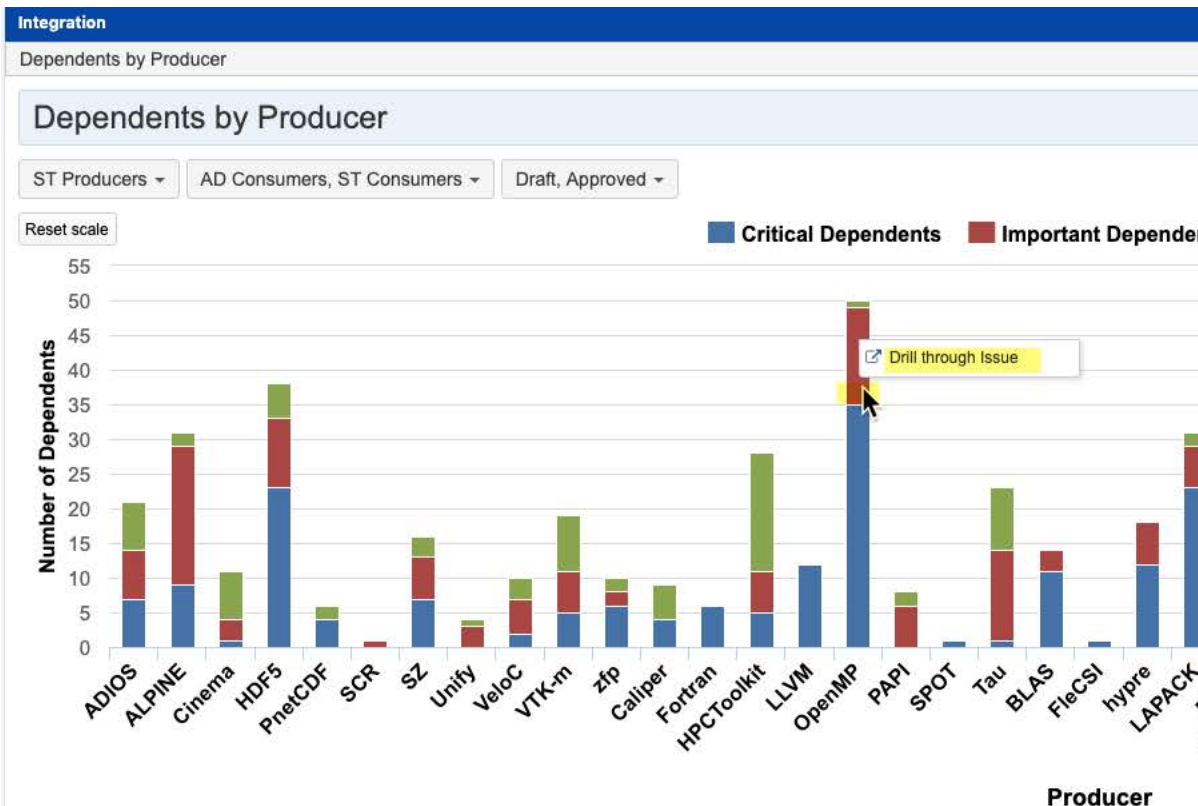
Drop files to attach, or [browse](#).

12

Labels

auto-generated × ▾

Begin typing to find and create labels or press down to select a suggested label.



ECP SW Technology Software Architecture – Spack Software Distribution System



Spack enables Software distribution for HPC

- Spack automates the build and installation of scientific software
- Packages are *templated*, so that users can easily tune for the host environment

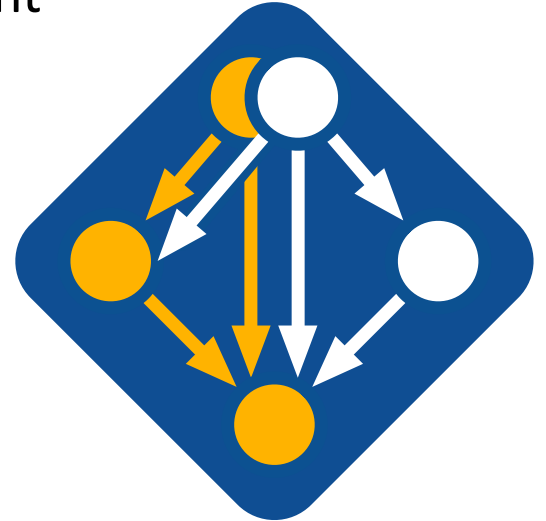
No installation required: clone and go

```
$ git clone https://github.com/spack/spack
$ spack install hdf5
```

Simple syntax enables complex installs

```
$ spack install hdf5@1.10.5
$ spack install hdf5@1.10.5 %clang@6.0
$ spack install hdf5@1.10.5 +threadssafe
$ spack install hdf5@1.10.5 cppflags="-O3 -g3"
$ spack install hdf5@1.10.5 target=haswell
$ spack install hdf5@1.10.5 +mpi ^mpich@3.2
```

- Ease of use of mainstream tools, with flexibility needed for HPC tuning
- Major victories:
 - ARES porting time on a new platform was reduced from **2 weeks to 3 hours**
 - Deployment time for 1,300-package stack on Summit supercomputer reduced from **2 weeks to a 12-hour overnight build**
 - Used by teams across ECP to **accelerate development**



github.com/spack/spack

Spack is being used on many of the top HPC systems

- Official deployment tool for the U.S. Exascale Computing Project
- 7 of the top 10 supercomputers
- High Energy Physics community
 - Fermilab, CERN, collaborators
- Astra (Sandia)
- Fugaku (Japanese National Supercomputer Project)



Fugaku coming to RIKEN in 2021
DOE/MEXT collaboration



Summit (ORNL), Sierra (LLNL)

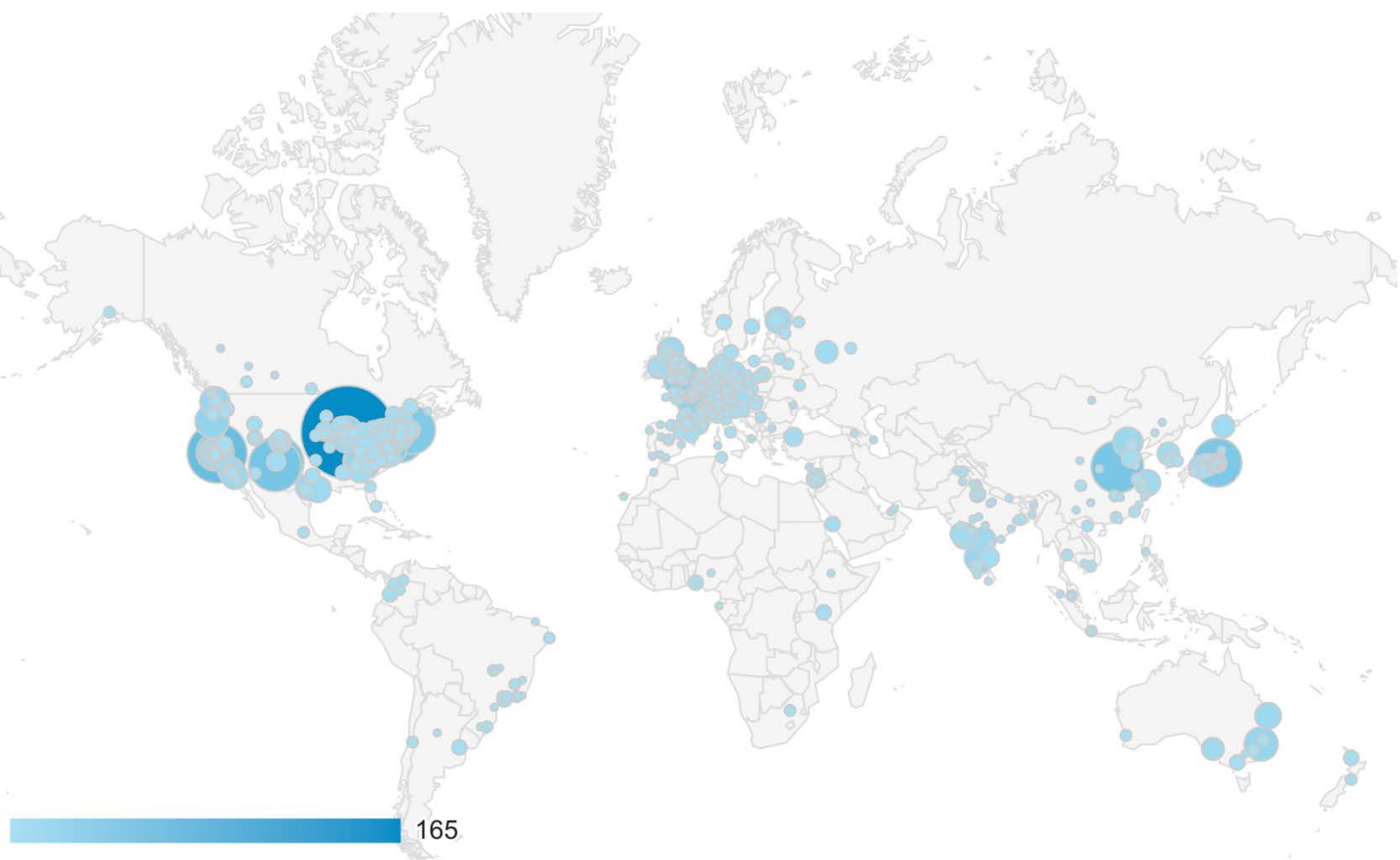


SuperMUC-NG (LRZ,
Germany)



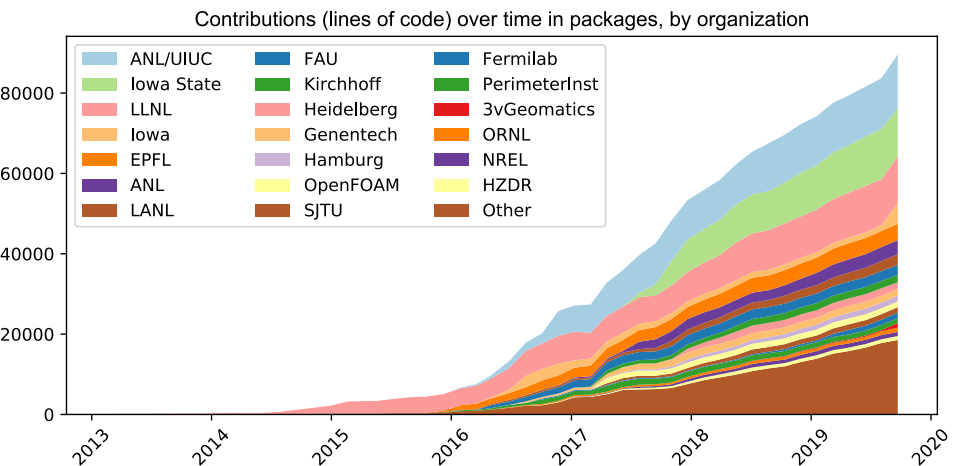
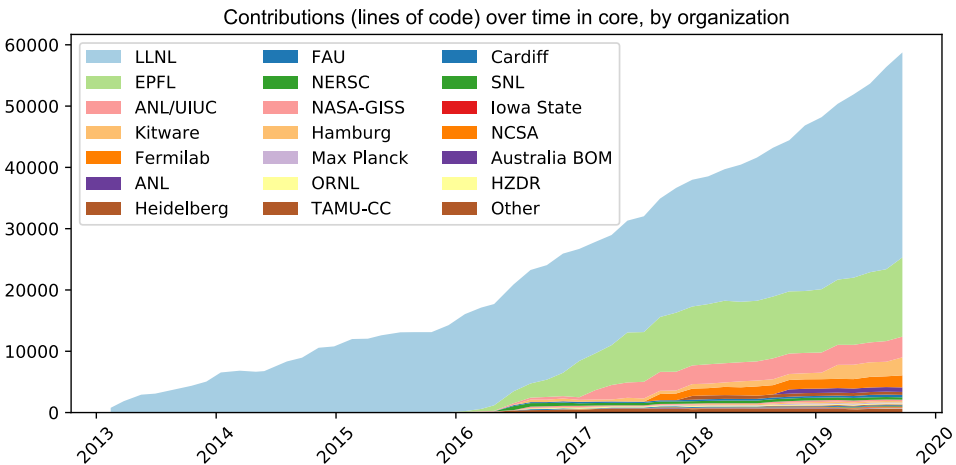
Edison, Cori, Perlmutter (NERSC)

Spack is used worldwide!



Active users of Spack documentation site for one month
<https://spack.readthedocs.io>

Over **3,400** software packages
Over **2,000** monthly active users
Over **400** contributors (and growing)



Spack strategy is to enable exascale software distribution on *both* bare metal and containers

1. New capabilities to make HPC packaging easy and automated

- Optimized builds and package binaries that exploit the hardware
- Workflow automation for facilities, developers, and users
- Strong integration with containers as well as facility deployments

2. Develop exascale enhancements to container runtimes

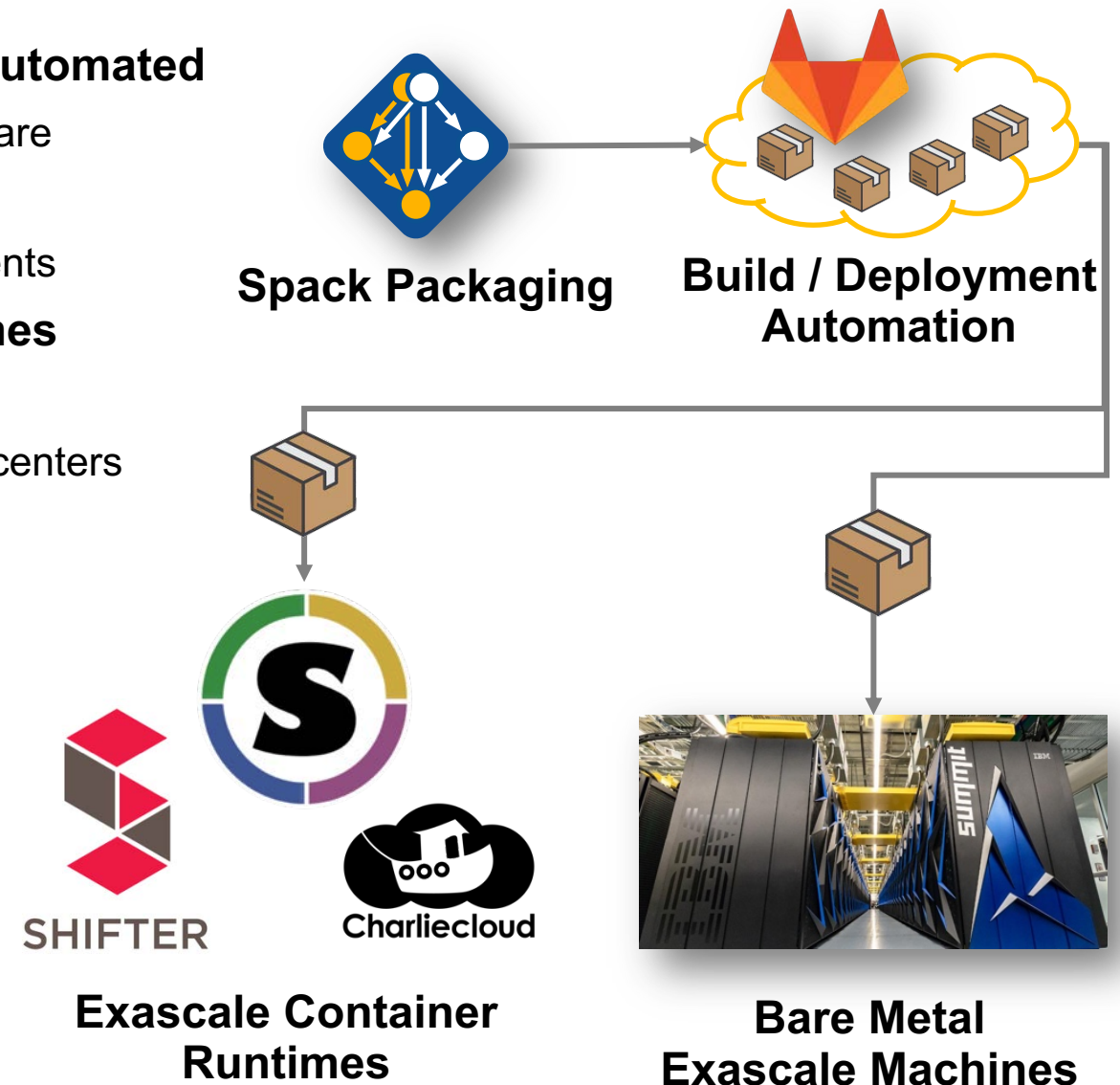
- Understand and automate performance/portability tradeoffs
- Optimized containers & build automation for exascale HPC centers
- Enable portability from laptops to exascale

3. Outreach to users

- Ongoing working groups, Best practice guides
- Tutorials, workshops, BOFs for Spack and Containers

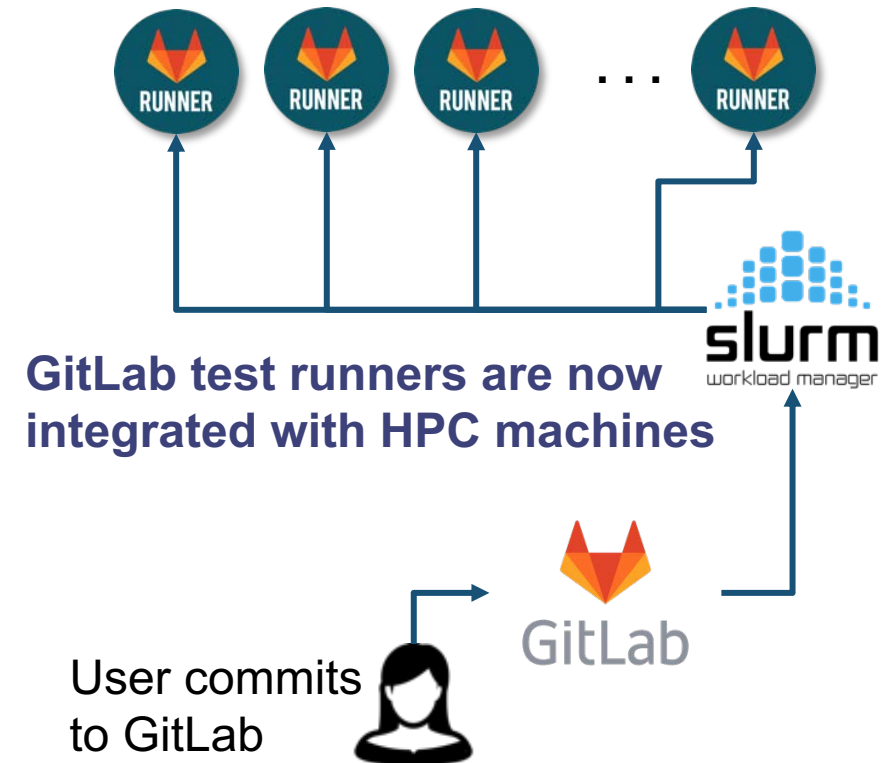
4. Collaboration across ECP

- Work with H1 and other areas to build service infrastructure
- Facilitate curation of packages through E4S and facilities
- Ongoing ECP user support



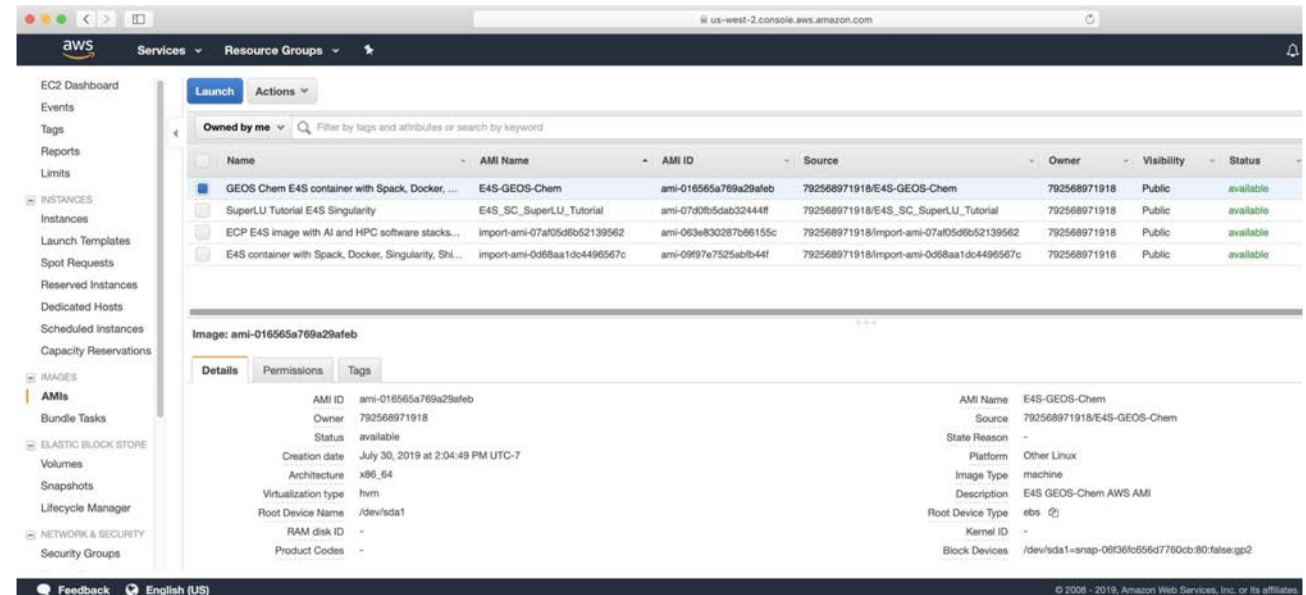
Spack heavily involved in the ECP CI project.

- We have added security features to the open source GitLab product.
 - Integration with center identity management
 - Integration with schedulers like SLURM, LSF
- We are democratizing testing at Livermore Computing
 - Users can run tests across 30+ machines by editing a file
 - Previously, each team had to administer own servers
- ECP sites are deploying GitLab CI for users
 - All HPC centers can leverage these improvements
 - NNSA labs plan to deploy common high-side CI infrastructure
 - We are developing new security policies to allow external open source code to be tested safely on key machines



Integration and Interoperability: E4S on AWS

- E4S AWS public image ami-063e830287b86155c (US-West-2 Oregon) has following container runtimes:
 - Docker
 - Shifter
 - Singularity
 - Charliecloud
- Spack with base PMR components
- E4S full featured Singularity image
 - (exascaleproject/sdk:AHM19)
- Used in ISC-HPC 2019 tutorials
- **Used as base image for NASA GEOS-Chem E4S public image**
- Resources provided by AWS AI/ML team



Reproducible, Customizable Container Builds & Spack Mirrors

- E4S provides base images and recipes for building Docker containers based on SDKs
 - Git: <https://github.com/UO-OACISS/e4s>
 - Base images released (September 2019):
 - UBI 7.6 (RHEL Universal Binary Image for container builds) for x86_64
 - Centos 7.6 for x86_64
 - Ubuntu 18.04 for x86_64
 - UBI 7.6 (RHEL) for ppc64le
- E4S provides **build caches for Spack for native bare-metal as well as container builds based installation** of ST products
 - Build caches: <https://oaciss.uoregon.edu/e4s>
 - **The build cache model can be extended to target platforms**, and can be managed by facilities staff when appropriate.

E4S Build Cache Binaries

https://oaciss.uoregon.edu/e4s/inventory.html

Spack E4S Build Cache

Last updated: 20-Sep-2019 11:07 PST

Click on one of the packages below to see a list of all available variants.

1148 Spack binaries in the build cache

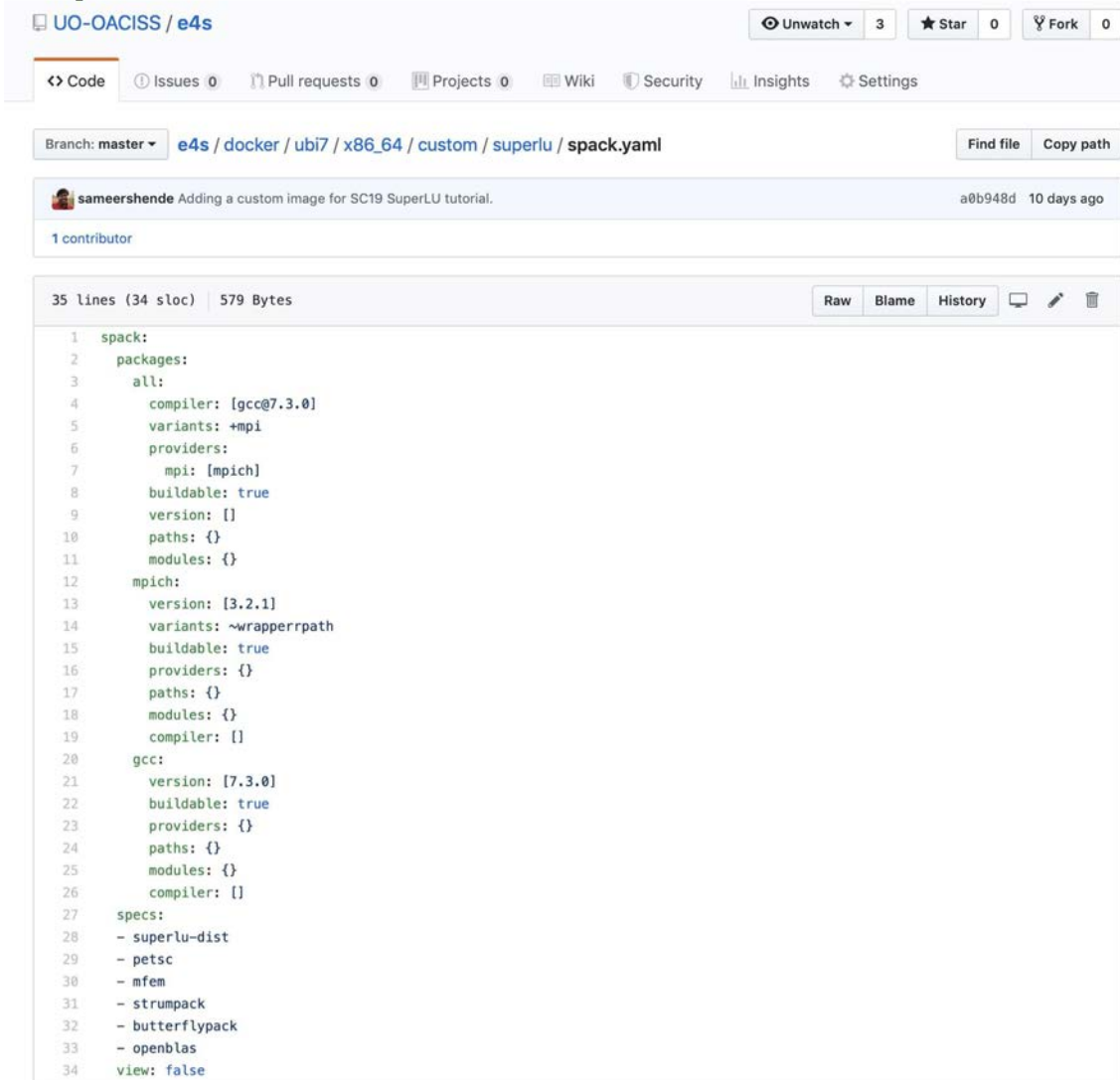
adios2@2.4.0 aml@0.1.0 argobots@1.0rc1 arpack-ng@3.7.0 autoconf@2.69 automake@1.16.1 axl@0.1.1 bdfpc@1.0.5 binutils@2.32 bison@3.0.5 bmi@develop
bolt@1.0rc1 boost@1.70.0 butterflypack@1.0.1 bzip2@1.0.8 c-blosc@1.17.0 cairo@1.16.0 caliper@2.0.1 cinch@develop cmake@3.15.1 cmake@3.15.3
cuda@10.0.130 cuda@10.1.243 curl@7.63.0 darshan-runtime@3.1.7 darshan-util@3.1.7 diffutils@3.7 double-conversion@2.0.1 dtcmp@1.1.0
dyninst@10.1.0

Click on the full spec link to find out more.

Link	Arch	OS	Compiler	Created	Full Hash
Full Spec	x86_64	centos7	gcc@7.3.0	18-Sep-2019 19:07	m46bcmvfkvly5iz5iygg4mmta7myiers
Full Spec	x86_64	centos7	gcc@7.3.0	18-Sep-2019 19:11	v2wfu3g3n7x4gndevks2vblmge53qs7o
Full Spec	x86_64	rhel7	gcc@7.3.0	15-Sep-2019 22:08	cwadhzs6dnelh5dw2kvihgti5477uxjv
Full Spec	x86_64	rhel7	gcc@7.3.0	15-Sep-2019 22:13	nokyntwy4poe4ips3pqlam2sft5s7sk
Full Spec	x86_64	ubuntu18.04	gcc@7.3.0	18-Sep-2019 19:16	lke6kpqlc5tpwxnwlthstek3wpefydy
Full Spec	x86_64	ubuntu18.04	gcc@7.3.0	18-Sep-2019 19:21	e25zc3763g75iaas5wwbsfnuxjybjfz

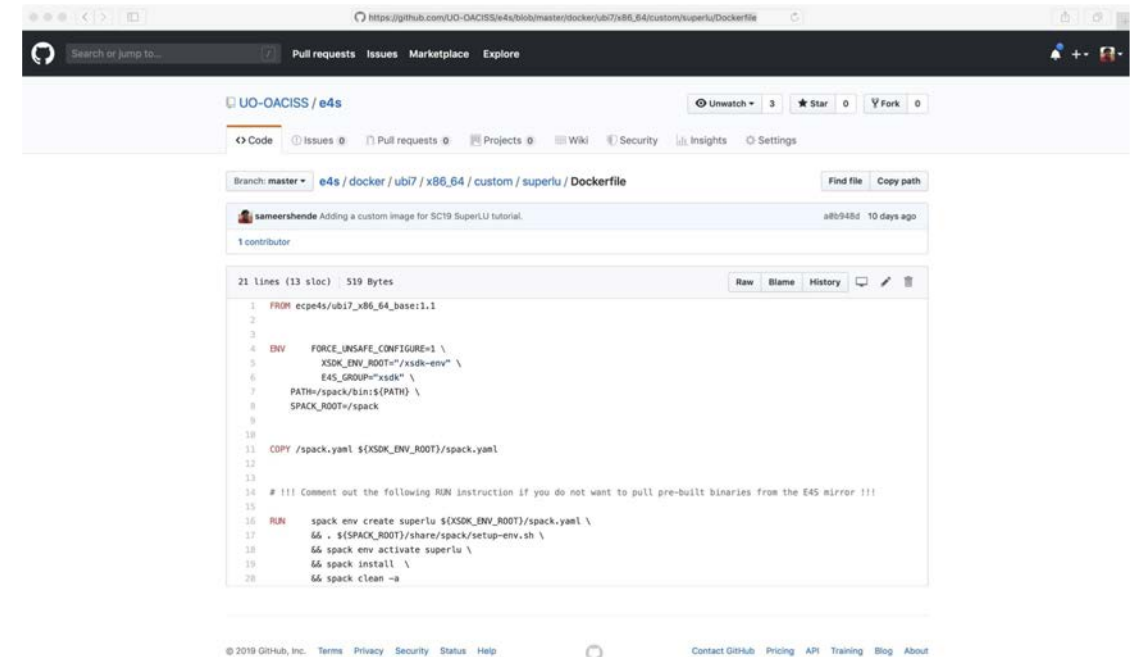
elfutils@0.176 environment-modules@4.3.0 er@0.0.3 expat@2.2.5 faodel@1.1906.1 findutils@4.6.0 flatcc@0.5.3 flecsi@develop flex@2.6.4 font-util@1.3.2
fontconfig@2.12.3 fontproto@2.1.3 freetype@2.9.1 gasnet@2019.3.0 gcc@7.3.0 gdbm@1.18.1 gettext@0.19.8.1 git@2.21.0 glib@2.56.3 glm@0.9.7.1
glproto@1.4.17 gmp@6.1.2 googletest@1.8.1 gotcha@0.0.2 gotcha@1.0.2 gperf@3.0.4 harfbuzz@2.3.1 hdf5@1.10.5 help2man@1.47.8 hpctoolkit@2019.08.14
hwloc@1.11.11 hypre@2.16.0 hypre@2.17.0 icu4c@64.1 inputproto@2.3.2 intel-mkl@2019.3.199 intel-tbb@2019.4 intel-xed@2019.03.01 isl@0.18 jsoncpp@1.9.1

Reproducible Container Builds using E4S Base Images



The screenshot shows the GitHub repository 'UO-OACISS / e4s'. The file 'spack.yaml' is selected, showing 35 lines of code. The code defines a Spack environment for building SuperLU. It includes a 'packages' section with 'all' and 'mpi' packages, and a 'specs' section listing the packages to be built: 'superlu-dist', 'petsc', 'mfem', 'strumpack', 'butterflypack', 'openblas', and 'view: false'.

```
1 spack:
2   packages:
3     all:
4       compiler: [gcc@7.3.0]
5       variants: +mpi
6       providers:
7         mpi: [mpich]
8       buildable: true
9       version: []
10      paths: {}
11      modules: {}
12    mpich:
13      version: [3.2.1]
14      variants: ~wrapperrpath
15      buildable: true
16      providers: {}
17      paths: {}
18      modules: {}
19      compiler: []
20    gcc:
21      version: [7.3.0]
22      buildable: true
23      providers: {}
24      paths: {}
25      modules: {}
26      compiler: []
27    specs:
28      - superlu-dist
29      - petsc
30      - mfem
31      - strumpack
32      - butterflypack
33      - openblas
34    view: false
```



The screenshot shows the GitHub repository 'UO-OACISS / e4s'. The file 'Dockerfile' is selected, showing 21 lines of code. The Dockerfile defines a container build for SuperLU. It starts with the base image 'FROM e4s/ubi7_x86_64_base:1.1'. It then sets environment variables for 'FORCE_UNSAFE_CONFIGURE', 'XSDK_ENV_ROOT', 'E4S_GROUP', 'PATH', and 'SPACK_ROOT'. It copies the 'spack.yaml' file to the container. Finally, it runs a command to create the SuperLU environment and install the packages.

```
1 FROM e4s/ubi7_x86_64_base:1.1
2
3
4 ENV FORCE_UNSAFE_CONFIGURE=1 \
5     XSDK_ENV_ROOT="/xsdk-env" \
6     E4S_GROUP="xsdk" \
7     PATH="/spack/bin:${PATH}" \
8     SPACK_ROOT="/spack"
9
10 COPY /spack.yaml ${XSDK_ENV_ROOT}/spack.yaml
11
12 # !!! Comment out the following RUN instruction if you do not want to pull pre-built binaries from the E4S mirror !!!
13
14 RUN spack env create superlu ${XSDK_ENV_ROOT}/spack.yaml \
15     && . ${SPACK_ROOT}/share/spack/setup-env.sh \
16     && spack env activate superlu \
17     && spack install \
18     && spack clean -a
```

- PMR SDK base image (UBI 7.6) has Spack build cache mirror and GPG key installed.
- Base image has GCC and MPICH configured for MPICH ABI level replacement (with system MPI).
- Customized container build using binaries from E4S Spack build cache for fast deployment.
- No need to rebuild packages from the source code.
- Same recipe for container and native bare-metal builds with Spack!

Spack Build Caches from E4S Base Images

Index of /e4s/x86_64/build_cache/linux-rhel7-x86_64/gcc-7.3.0

Name	Last modified	Size	Description
Parent Directory		-	
adios2-2.4.0/	19-Sep-2019 06:25	-	
arpack-ng-3.7.0/	12-Sep-2019 12:35	-	
autoconf-2.69/	17-Sep-2019 17:24	-	
automake-1.16.1/	17-Sep-2019 17:31	-	
axi-0.1.1/	19-Sep-2019 06:28	-	
binutils-2.32/	15-Sep-2019 21:57	-	
bison-3.0.5/	28-Aug-2019 08:17	-	
boost-1.70.0/	19-Sep-2019 06:16	-	
butterflypack-1.0.1/	12-Sep-2019 12:35	-	
bzip2-1.0.8/	17-Sep-2019 17:47	-	
c-blosc-1.17.0/	27-Aug-2019 09:04	-	
caliper-2.0.1/	06-Sep-2019 08:41	-	
cinch-develop/	28-Aug-2019 18:57	-	
cmake-3.15.1/	17-Sep-2019 17:57	-	
cmake-3.15.3/	18-Sep-2019 13:17	-	
cuda-10.0.130/	05-Sep-2019 12:59	-	
cuda-10.1.243/	11-Sep-2019 13:42	-	
curl-7.63.0/	17-Sep-2019 18:03	-	
darshan-runtime-3.1.7/	19-Sep-2019 06:22	-	
darshan-util-3.1.7/	27-Aug-2019 09:02	-	
diffutils-3.7/	18-Sep-2019 14:23	-	
dtcomp-1.1.0/	19-Sep-2019 06:28	-	
dynamat-10.1.0/	15-Sep-2019 22:13	-	
elfutils-0.176/	15-Sep-2019 21:58	-	
environment-modules-4.3.0/	17-Sep-2019 17:59	-	
er-0.0.3/	19-Sep-2019 06:29	-	
expat-2.2.5/	17-Sep-2019 17:24	-	
faodel-1.1986.1/	19-Sep-2019 06:27	-	
findutils-4.6.0/	17-Sep-2019 17:24	-	
flecsi-develop/	18-Sep-2019 14:44	-	
flex-2.6.4/	28-Aug-2019 08:17	-	
fontconfig-2.9.1/	27-Aug-2019 11:48	-	
gannet-2019.3.0/	27-Aug-2019 10:18	-	
gdbm-1.18.1/	17-Sep-2019 17:47	-	
gettext-0.19.8.1/	17-Sep-2019 17:26	-	
git-2.21.0/	17-Sep-2019 18:03	-	
glm-0.9.7.1/	29-Aug-2019 12:43	-	
glproto-1.4.17/	27-Aug-2019 11:43	-	
googletest-1.8.1/	27-Aug-2019 09:09	-	
gotcha-1.0.2/	10-Sep-2019 13:01	-	
hdf5-1.10.5/	19-Sep-2019 06:21	-	
help2man-1.47.8/	28-Aug-2019 08:17	-	
hpc-toolkit-2019.08.14/	15-Sep-2019 22:17	-	
hwloc-1.11.1/	15-Sep-2019 18:27	-	
hypre-2.16.0/	17-Aug-2019 09:08	-	
hypre-2.17.0/	29-Aug-2019 12:55	-	
inputproto-2.3.2/	27-Aug-2019 11:48	-	
intel-mkl-2019.3.199/	05-Sep-2019 13:09	-	
intel-tbb-2019.4/	10-Sep-2019 12:49	-	
intel-xed-2019.03.01/	10-Sep-2019 12:57	-	
kbrproto-1.0.7/	28-Aug-2019 08:27	-	
kokkos-2.8.0/	10-Aug-2019 07:35	-	
kokkos-2.9.0/	27-Aug-2019 10:16	-	
kvtree-1.0.2/	19-Sep-2019 06:25	-	
legion-19.06.0/	27-Aug-2019 10:19	-	
libbsd-0.9.1/	17-Sep-2019 17:30	-	
libdwarf-20180129/	10-Sep-2019 12:57	-	
libedit-3.1-20170329/	27-Aug-2019 11:48	-	
libelf-0.8.13/	10-Sep-2019 12:48	-	
libfabric-1.8.0/	27-Aug-2019 09:04	-	
libffi-3.2.1/	05-Sep-2019 13:15	-	
libiberty-2.31.1/	10-Sep-2019 12:52	-	
libice-1.0.9/	27-Aug-2019 11:48	-	
libiconv-1.15/	17-Sep-2019 17:59	-	
libjpeg-turbo-2.0.2/	28-Aug-2019 08:28	-	
libmonitor-2018.07.18/	10-Sep-2019 12:48	-	
libpciaccess-0.13.5/	17-Sep-2019 17:30	-	
libpm4-4.10.1/	10-Sep-2019 13:02	-	
libpng-1.6.34/	27-Aug-2019 09:03	-	
libpthread-stubs-0.4/	27-Aug-2019 09:03	-	
libquo-1.3/	27-Aug-2019 10:16	-	
libsigsegv-2.11/	17-Sep-2019 17:54	-	
libsm-1.2.2/	27-Aug-2019 11:43	-	
libsodium-1.0.17/	27-Aug-2019 09:02	-	

- x86_64 build cache
 - 914 binaries
 - 40 GB on disk

Index of /e4s/ppc64le/build_cache/linux-centos7-ppc64le/gcc-7.3.0

Name	Last modified	Size	Description
Parent Directory		-	
adios2-2.4.0/	20-Sep-2019 09:38	-	
aml-0.1.0/	20-Sep-2019 11:03	-	
argobots-1.0rc1/	20-Sep-2019 09:44	-	
autoconf-2.69/	18-Sep-2019 12:31	-	
automake-1.16.1/	18-Sep-2019 12:23	-	
axi-0.1.1/	20-Sep-2019 09:38	-	
bmi-devel/	20-Sep-2019 09:37	-	
boost-1.70.0/	20-Sep-2019 09:43	-	
bzip2-1.0.8/	18-Sep-2019 12:31	-	
c-blosc-1.17.0/	20-Sep-2019 09:44	-	
cmake-3.15.3/	18-Sep-2019 12:25	-	
curl-7.63.0/	18-Sep-2019 12:31	-	
darshan-runtime-3.1.7/	20-Sep-2019 09:37	-	
darshan-util-3.1.7/	20-Sep-2019 09:39	-	
diffutils-3.7/	18-Sep-2019 12:31	-	
dtcomp-1.1.0/	20-Sep-2019 09:37	-	
environment-modules-4.3.0/	18-Sep-2019 12:30	-	
er-0.0.3/	20-Sep-2019 09:37	-	
expat-2.2.5/	18-Sep-2019 12:24	-	
findutils-4.6.0/	18-Sep-2019 12:23	-	
flatcc-0.5.3/	20-Sep-2019 09:37	-	
gasnet-2019.3.0/	20-Sep-2019 11:05	-	
gdbm-1.18.1/	18-Sep-2019 12:35	-	
gettext-0.19.8.1/	18-Sep-2019 12:35	-	
git-2.21.0/	18-Sep-2019 12:33	-	
gotcha-0.0.2/	20-Sep-2019 09:44	-	
hdf5-1.10.5/	20-Sep-2019 11:03	-	
hwloc-1.11.1/	20-Sep-2019 11:02	-	
kokkos-2.9.0/	20-Sep-2019 11:02	-	
kvtree-1.0.2/	20-Sep-2019 09:38	-	
legion-19.06.0/	20-Sep-2019 11:03	-	
leveldb-1.22/	20-Sep-2019 09:37	-	
libbsd-0.9.1/	18-Sep-2019 12:36	-	
libfabric-1.8.0/	20-Sep-2019 09:37	-	
libiconv-1.15/	18-Sep-2019 12:31	-	
libpciaccess-0.13.5/	18-Sep-2019 12:31	-	
libpng-1.6.34/	20-Sep-2019 09:41	-	
libpthread-stubs-0.4/	20-Sep-2019 09:44	-	
libquo-1.3/	20-Sep-2019 11:03	-	
libsigsegv-2.11/	18-Sep-2019 12:31	-	
libsodium-1.0.17/	20-Sep-2019 09:37	-	
libtool-2.4.6/	18-Sep-2019 12:22	-	
libxml2-2.9.9/	18-Sep-2019 12:30	-	

- IBM Power 9 (ppc64le) build cache
 - 101 binaries
 - 2.6 GB on disk
 - early stages of effort
 - Initial ARM 64 build cache is underway

ECP SW Technology Software Architecture – SDKs



Software Development Kits (SDKs): Key delivery vehicle for ECP

A collection of related software products (packages) where coordination across package teams improves usability and practices, and foster community growth among teams that develop similar and complementary capabilities

- **Domain scope**

Collection makes functional sense

- **Interaction model**

How packages interact; compatible, complementary, interoperable

- **Community policies**

Value statements; serve as criteria for membership

- **Meta-infrastructure**

Invokes build of all packages (Spack), shared test suites

- **Coordinated plans**

Inter-package planning. Augments autonomous package planning

- **Community outreach**

Coordinated, combined tutorials, documentation, best practices

ECP ST SDKs: Grouping similar products for collaboration & usability

Programming Models &
Runtimes Core

Tools & Technologies

Compilers & Support

Math Libraries (xSDK)

Viz Analysis and Reduction

Data mgmt., I/O Services & Checkpoint/
Restart



“Unity in essentials, otherwise diversity”



xSDK community policies

<https://xsdk.info/policies>

xSDK compatible package: Must satisfy mandatory xSDK policies:

- M1.** Support xSDK community GNU Autoconf or CMake options.
- M2.** Provide a comprehensive test suite.
- M3.** Employ user-provided MPI communicator.
- M4.** Give best effort at portability to key architectures.
- M5.** Provide a documented, reliable way to contact the development team.
- M6.** Respect system resources and settings made by other previously called packages.
- M7.** Come with an open source license.
- M8.** Provide a runtime API to return the current version number of the software.
- M9.** Use a limited and well-defined symbol, macro, library, and include file name space.
- M10.** Provide an accessible repository (not necessarily publicly available).
- M11.** Have no hardwired print or IO statements.
- M12.** Allow installing, building, and linking against an outside copy of external software.
- M13.** Install headers and libraries under <prefix>/include/ and <prefix>/lib/.
- M14.** Be buildable using 64 bit pointers. 32 bit is optional.
- M15.** All xSDK compatibility changes should be sustainable.
- M16.** The package must support production-quality installation compatible with the xSDK install tool and xSDK metapackage.

Also **recommended policies**, which currently are encouraged but not required:

- R1.** Have a public repository.
- R2.** Possible to run test suite under valgrind in order to test for memory corruption issues.
- R3.** Adopt and document consistent system for error conditions/exceptions.
- R4.** Free all system resources it has acquired as soon as they are no longer needed.
- R5.** Provide a mechanism to export ordered list of library dependencies.
- R6.** Provide versions of dependencies.

xSDK member package: Must be an xSDK-compatible package, *and* it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

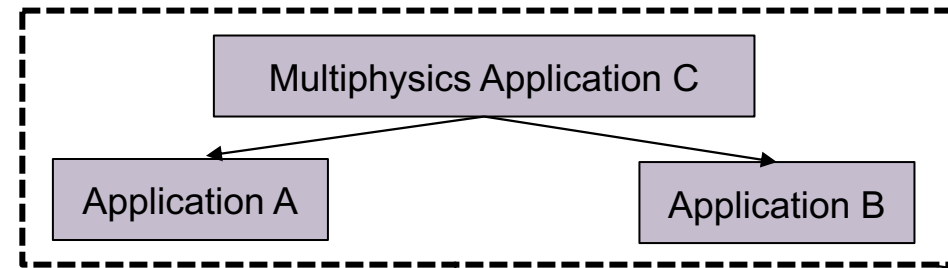
We welcome feedback. What policies make sense for your software?



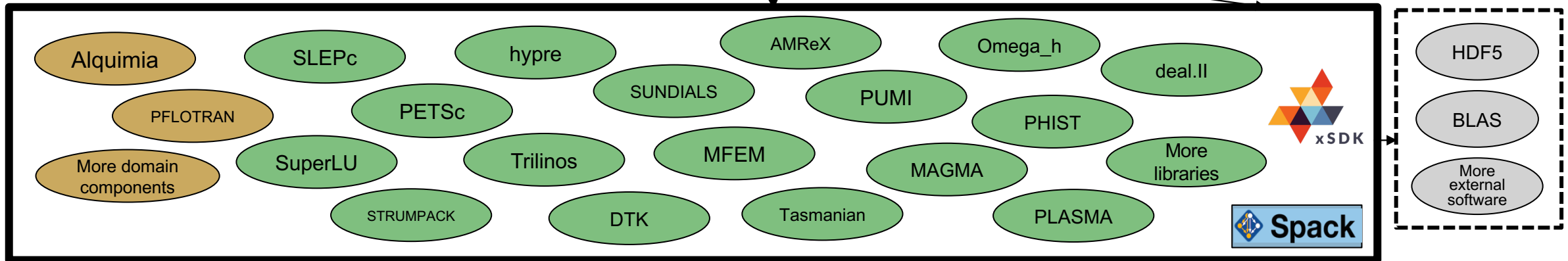
xSDK version 0.4.0: December 2018

<https://xsdk.info>

Each xSDK member package uses or can be used with one or more xSDK packages, and the connecting interface is regularly tested for regressions.

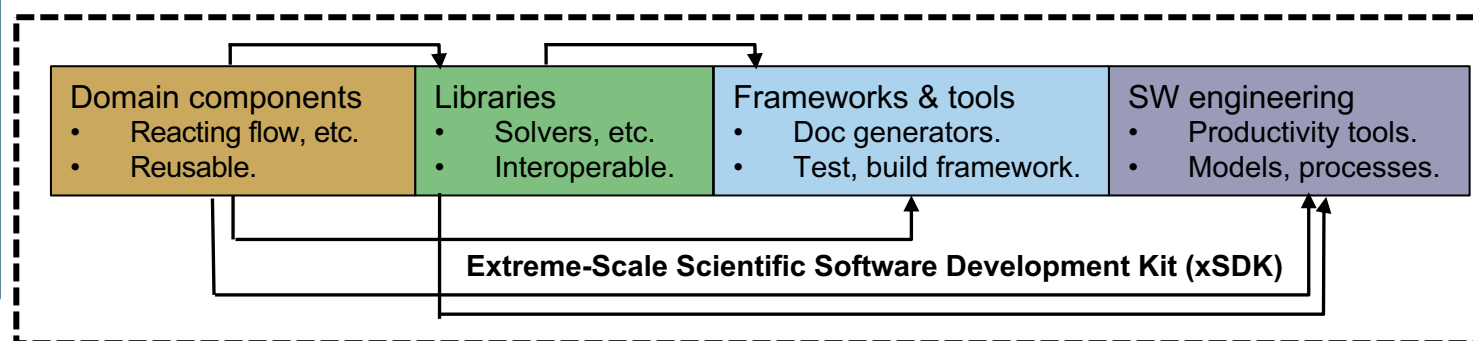


Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X



December 2018

- 17 math libraries
- 2 domain components
- 16 mandatory xSDK community policies
- Spack xSDK installer

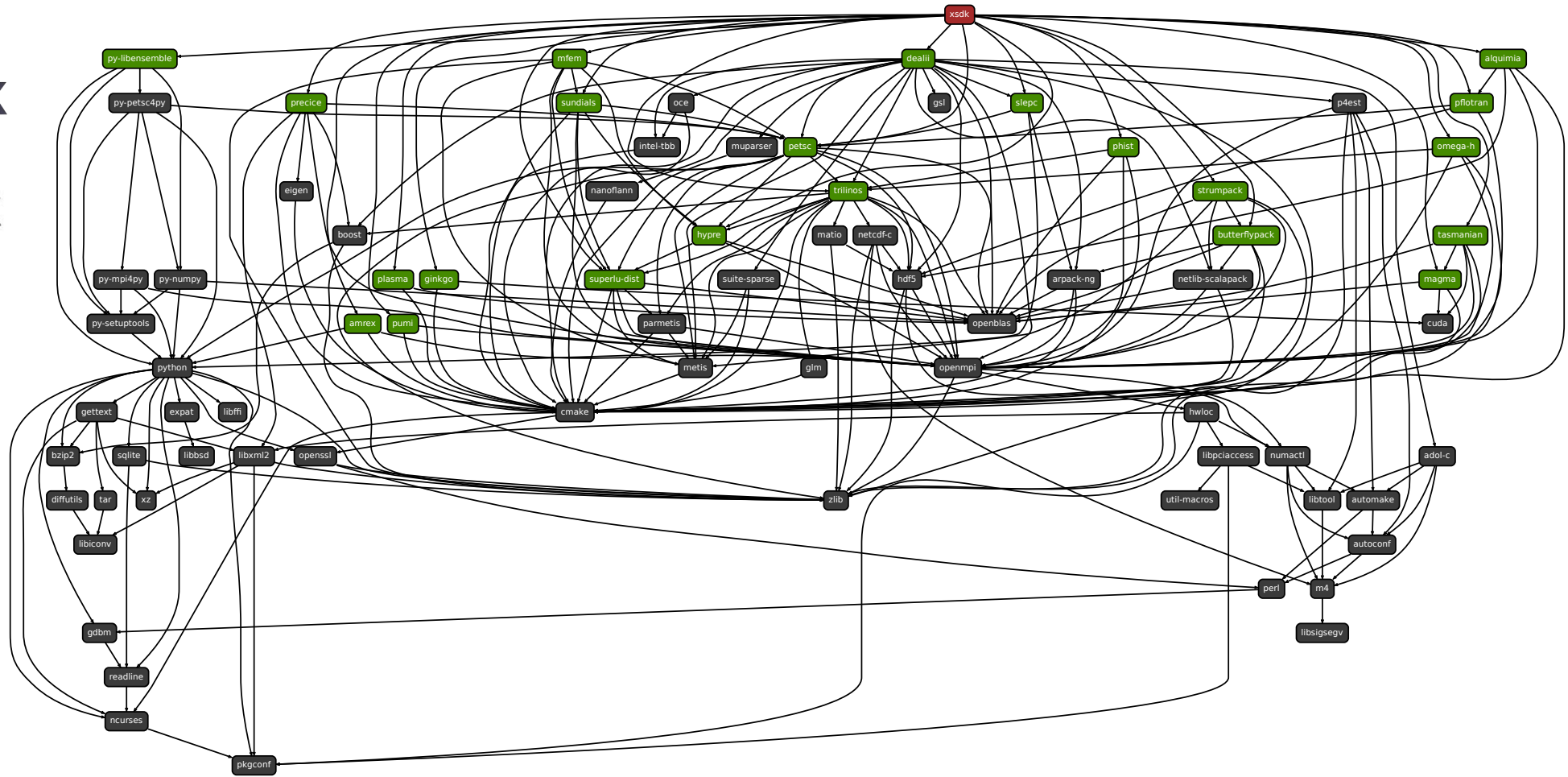


Impact: Improved code quality, usability, access, sustainability

Foundation for work on performance portability, deeper levels of package interoperability



Dependency



xSDK version 0.5: November 2019

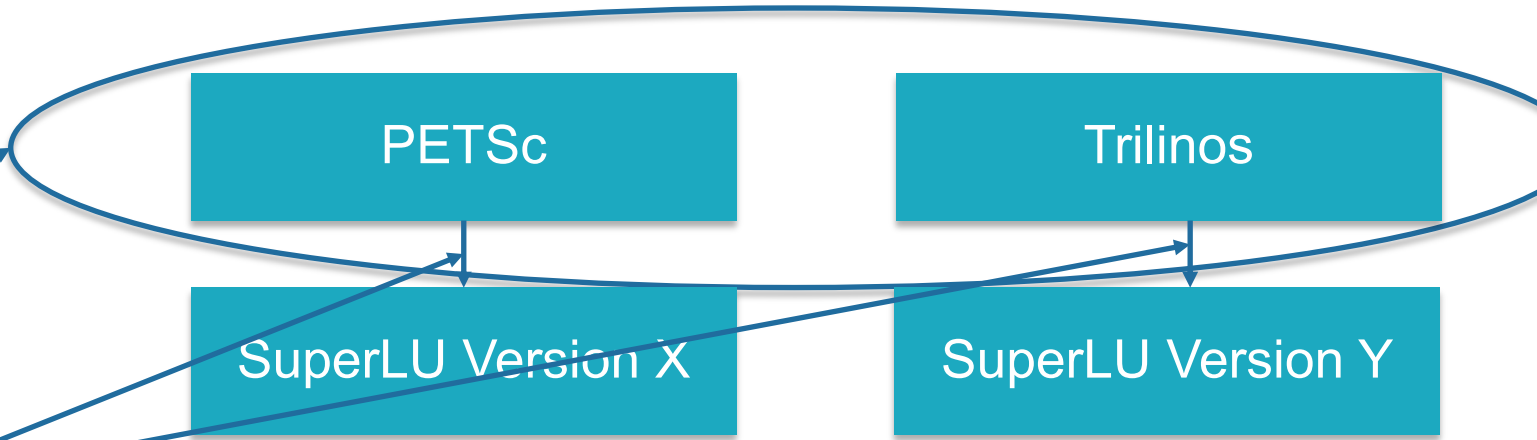
(21 math libs, 2 domain-specific packages)

- AMReX
- ButterflyPACK
- DTK
- deal.ii
- Ginkgo
- hypre
- libEnsemble
- MAGMA
- MFEM
- Omega_h
- PETSc/TAO
- PHIST
- PLASMA
- preCICE
- PUMI
- SLEPc
- STRUMPACK
- SUNDIALS
- SuperLU
- Tasmanian
- Trilinos
- Pflotran
- Alquimia

Notes:

- Growth:
 - 5 in release 0.1.
 - 7 in 0.2
 - 9 in 0.3
 - 19 in 0.4
 - 23 in 0.5
- You do not need to build all packages.
- We build and test all packages.
- Any subset is guaranteed to build if using the same build parameters, platforms.
- Similar builds should work or require less effort for success.

SDK “Horizontal” Grouping: Key Quality Improvement Driver



Horizontal (vs Vertical) Coupling

- Common substrate
- Similar function and purpose
 - e.g., compiler frameworks, math libraries
- Potential benefit from common Community Policies
 - Best practices in software design and development and customer support
- Used together, but not in the long vertical dependency chain sense
- Support for (and design of) common interfaces
 - Commonly an aspiration, not yet reality

Horizontal grouping:

- Assures $X=Y$.
- Protects against regressions.
- Transforms code coupling from heroic effort to turnkey.

Development of xSDK Community Policies

- Community policies now available on Github for revision control and to preserve history:
<https://github.com/xsdk-project/xsdk-community-policies>
- Established process on how to change or add policies
- Newest release: 0.5.0

Also **recommended policies**, which currently are encouraged but not required:

- R1.** Have a public repository.
- R2.** Possible to run test suite under valgrind in order to test for memory corruption issues.
- R3.** Adopt and document consistent system for error conditions/exceptions.
- R4.** Free all system resources it has acquired as soon as they are no longer needed.
- R5.** Provide a mechanism to export ordered list of library dependencies.
- R6.** Provide versions of dependencies.
- R7.** Have README, SUPPORT, LICENSE, and CHANGELOG file in top directory.

Changes in 0.5.0:

- ❑ **New recommended policy R7**
- ❑ **Dropped the requirement to detect MPI 2 features in M3**
- ❑ **Made various editorial changes in M5, M13, M15, and R2 for clarification or to fix typos.**

We welcome feedback. What policies make sense for your software?

<https://xsdk.info/policies>

xSDK compatible package: Must satisfy mandatory xSDK policies:

- M1.** Support xSDK community GNU Autoconf or CMake options.
- M2.** Provide a comprehensive test suite.
- M3.** Employ user-provided MPI communicator.
- M4.** Give best effort at portability to key architectures.
- M5.** Provide a documented, reliable way to contact the development team.
- M6.** Respect system resources and settings made by other previously called packages.
- M7.** Come with an open source license.
- M8.** Provide a runtime API to return the current version number of the software.
- M9.** Use a limited and well-defined symbol, macro, library, and include file name space.
- M10.** Provide an accessible repository (not necessarily publicly available).
- M11.** Have no hardwired print or IO statements.
- M12.** Allow installing, building, and linking against an outside copy of external software.
- M13.** Install headers and libraries under <prefix>/include/ and <prefix>/lib/.
- M14.** Be buildable using 64 bit pointers. 32 bit is optional.
- M15.** All xSDK compatibility changes should be sustainable.
- M16.** The package must support production-quality installation compatible with the xSDK install tool and xSDK metapackage.

Integration of Community Policies

- Potential new xSDK members will fill out compatibility form:
<https://github.com/xsdk-project/xsdk-policy-compatibility>
- xSDK team will check for compatibility and approve if compatible
- Designing and implementing xSDK policy checker to automate the policy compliance checking.
 - Implemented checkers for a few policy rules.
Examples:
 - M3: Employ user-provided MPI Communicator: Policy checker lists the file names that contain MPI_COMM_WORLD.
 - M7: Come with an open source license. Policy checker search license files at the top directly of the source and scan the headers of source code files.

xSDK Community Policy Compatibility for PETSc

This document summarizes the efforts of current and future xSDK member packages to achieve compatibility with the xSDK community policies. Below only short descriptions of each policy are provided. The full description is available [here](#) and should be considered when filling out this form.

Please, provide information on your compability status for each mandatory policy, and if possible also for recommended policies. If you are not compatible, state what is lacking and what are your plans on how to achieve compliance. For current xSDK member packages: If you were not compliant at some point, please describe the steps you undertook to fulfill the policy. This information will be helpful for future xSDK member packages.

Website: <https://www.mcs.anl.gov/petsc>

Mandatory Policies

Policy	Support	Notes
M1. Support xSDK community GNU Autoconf or CMake options.	Full	PETSc uses the GNU Autoconf options. The implementation is done with python code.
M2. Provide a comprehensive test suite for correctness of installation verification.	Full	PETSc has over 1000 test examples and a test harness that can execute the examples in parallel. It also collects information on the failures and can display them graphically, e.g., see ftp://ftp.mcs.anl.gov/pub/petsc/nightlylogs/archive/2017/09/19/master.html
M3. Employ userprovided MPI communicator (no MPI_COMM_WORLD).	Full	All PETSc objects take a MPI communicator in the constructor, allowing the user complete control over where each object exists and performs its computations.
M4. Give best effort at portability to new architectures (standard Linux)		

Processes for xSDK release and delivery

- **2-level release process**

- **xSDK**

- Ensure and test compatibility of mostly independent package releases

- **xSDK member packages**

- Achieve compatibility with xSDK community policies prior to release
 - <https://github.com/xsdk-project/xsdk-policy-compatibility>
 - Have a Spack package
 - Port to target platforms
 - Provide user support

- **Obtaining the latest release:** <https://xsdk.info/releases>

- **Draft xSDK package release process checklist:**

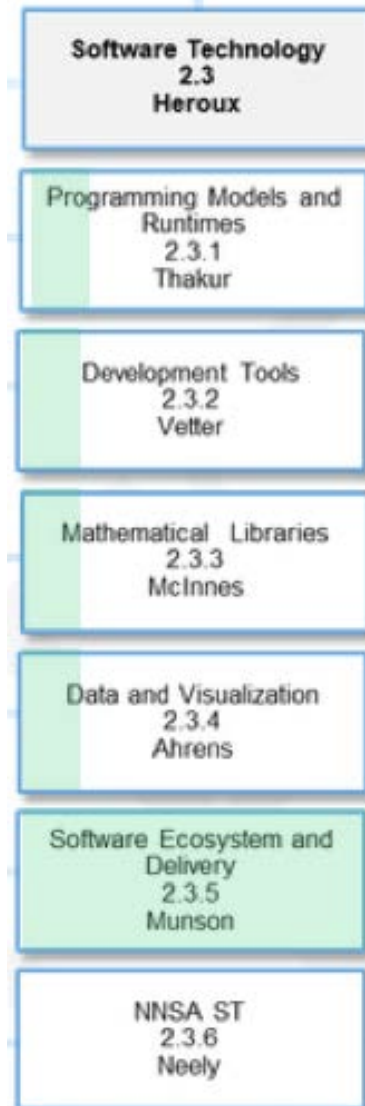
- <https://docs.google.com/document/d/16y2bL1RZg8wke0vY8c97ssvhRYNez34Q4QGg4LoIEUk/edit?usp=sharing>

xSDK delivery process

- Regular releases of software and documentation, primarily through member package release processes
- Anytime open access to production software from GitHub, BitBucket and related community platforms

There are 5 L4 projects to define and/or enhance SDKs

- Each L3 area has an L4 project devoted to SDK definition and coordination across the portfolio
- Software ecosystem L4 project focuses on packaging
 - Spack
 - Containers
- Strong coordination with H1 Software Deployment projects
- Drives milestone-based planning



ECP ST SDKs will span all technology areas

Motivation: Properly chosen cross-team interactions will build relationships that support interoperability, usability, sustainability, quality, and productivity within ECP ST.

Action Plan: Identify product groupings where coordination across development teams will improve usability and practices, and foster community growth among teams that develop similar and complementary capabilities.

PMR Core (17)	Compilers and Support (7)	Tools and Technology (11)	xSDK (16)	Visualization Analysis and Reduction (9)	Data mgmt, I/O Services, Checkpoint restart (12)	Ecosystem/E4S at-large (12)
QUO	openarc	TAU	hypre	ParaView	SCR	mpiFileUtils
Papyrus	Kitsune	HPCToolkit	FleSCI	Catalyst	FAODEL	TriBITS
SICM	LLVM	Dyninst Binary Tools	MFEM	VTK-m	ROMIO	MarFS
Legion	CHiLL autotuning comp	Gotcha	Kokkoskernels	SZ	Mercury (Mochi suite)	GUFI
Kokkos (support)	LLVM openMP comp	Caliper	Trilinos	zfp	HDF5	Intel GEOPM
RAJA	OpenMP V & V	PAPI	SUNDIALS	VisIt	Parallel netCDF	BEE
CHAI	Flang/LLVM Fortran comp	Program Database Toolkit	PETSc/TAO	ASCENT	ADIOS	FSEFI
PaRSEC*		Search (random forests)	libEnsemble	Cinema	Darshan	Kitten Lightweight Kernel
DARMA		Siboka	STRUMPACK	ROVER	UnifyCR	COOLR
GASNet-EX		C2C	SuperLU		VeloC	NRM
Qthreads		Sonar	ForTrilinos		IOSS	ArgoContainers
BOLT			SLATE		HXHIM	Spack
UPC++			MAGMA			
MPICH			DTK			
Open MPI			Tasmanian			
Umpire			TuckerMPI			
AML						

PMR

Tools

Math Libraries

Data and Vis

Ecosystems and delivery

Legend

Data SDK: HDF5 API Initiative

- Adding The HDF Group (THG) to the Data & Vis SDK project to assist in various aspects of improving the use of the HDF5 API
 - Support and training
 - Direct support and training will be provided to ECP applications and ST projects to assist in the performant use of HDF5
 - Compatible APIs with other I/O libraries
 - DataLib (pnetcdf) and ADIOS will develop interfaces compatible with the HDF5 API, possibly through the HDF5 Virtual Object Layer
 - THG will provide support to the I/O projects to assist in achieving this compatibility
- Both Kitware and THG will identify the I/O usage patterns w.r.t. metadata operations, bulk data operations, access patterns.
 - This will help identify usage patterns that can be improved on by applications and ST projects
 - Also will help identify usage patterns that HDF5 can work to optimize for.

SDK Summary

- SDKs will help reduce complexity of delivery:
 - Hierarchical build targets.
 - Distribution of software integration responsibilities.
- New Effort: Started in April 2018, fully established in August 2018.
- Extending the SDK approach to all ECP ST domains.
 - SDKs create a horizontal coupling of software products, teams.
 - Create opportunities for better, faster, cheaper – pick all three.
- First concrete effort: Spack target to build all packages in an SDK.
 - Decide on good groupings.
 - Not necessarily trivial: Version compatibility issues, Coordination of common dependencies.
- Longer term:
 - Establish community policies, enhance best practices sharing.
 - Provide a mechanism for shared infrastructure, testing, training, etc.
 - Enable community expansion beyond ECP.

E4S Documentation Initiative



ECP Software Product Documentation Goals

- Provide a single online location for *accurate* product descriptions for ECP software products.
- Derived requirements:
 - Sustainable: Must be integrated into software team workflows.
 - Incremental: Must build on community approaches to providing this kind of information.
 - Extensible: Must be usable by any open source software team.
- Strategy:
 - Use the open source community approach of specially-name files in software repositories.
 - Adopt commonly used file names when available.
 - Identify new information items not already being requested.
 - Develop new special file names for information beyond what is already captured.
 - Create web-based raking tool to capture information from product repositories and present in summary form on a webpage.

Status of efforts

- The math libraries “SDK” effort (xSDK) has developed a recommended policy for member product teams to provide the following files in the root directory of the default branch of the product repository:

- README[.md],
- SUPPORT[.md],
- LICENSE[.md],
- CHANGELOG[.md]

Branch: master xsdk-community-policies / package_policies / R7.md Find file Copy path

ulrikeyang Update R7.md 915368e on Jun 20

1 contributor

17 lines (9 sloc) 1.18 KB Raw Blame History

R7. It is recommended that each package should have a README or README.md, a SUPPORT or SUPPORT.md, a LICENSE or LICENSE.md, and a CHANGELOG or CHANGELOG.md file in its top directory. The README file should contain at least the following information:

- a **brief** description of the package
- information on how to install the package or a link to a file (e.g. INSTALL, INSTALL.md or similar) or a website with the installation information
- a link to the package webpage (if there is one)

The LICENSE file should contain the full text of the license. If the full text can be found in a different directory or via a link to a website it is sufficient to provide a link to the full text in the file.

The SUPPORT file should contain the contact information that is required by community policy M5 to be able to get help.

The CHANGELOG file should contain important changes or a link to a file or webpage with this information, but not each individual commit message.

If providing additional information, e.g. on how to contribute, authorship, code of conduct, etc, it is recommended to consider suggestions in <https://github.com/kmindi/special-files-in-repository-root/blob/master/README.md>

Next phase: Test process, determine other special files

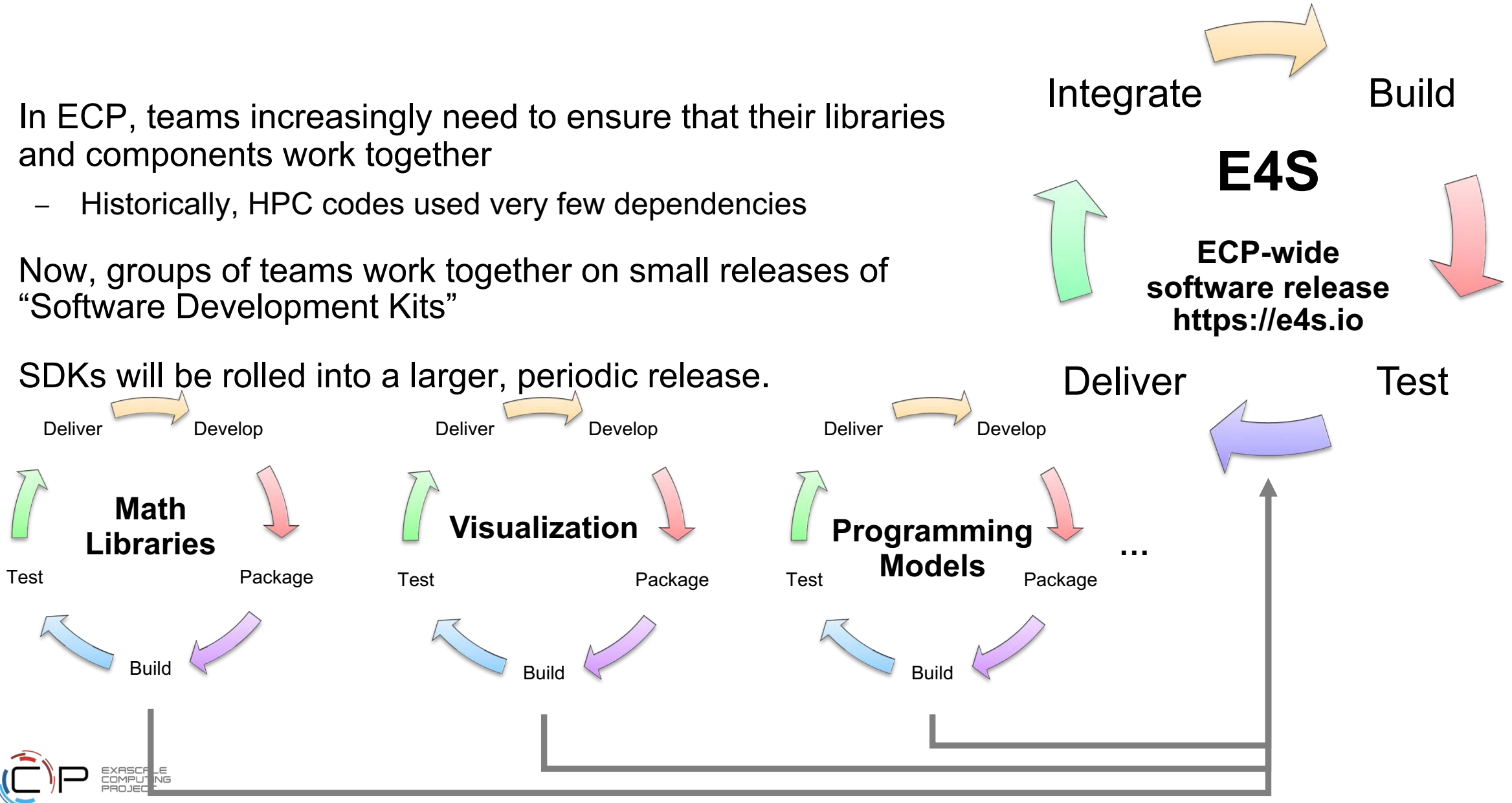
- Beyond the README, SUPPORT, LICENSE and CHANGELOG files, what information do we need from each product?
- Examples from <https://github.com/kmind/special-files-in-repository-root/blob/master/README.md> are:
 - CONTRIBUTING – how to contribute
 - CONTRIBUTORS – who contributed
 - ACKNOWLEDGMENTS
 - CODE_OF_CONDUCT
 - ISSUE_TEMPLATE, PULL_REQUEST_TEMPLATE
 - CODEOWNERS

Some recent new ideas for expansion

- Structure requirements for special files, e.g., README has a certain structure.
- README has status panels: CI passing, etc.
- Pointer to Spack recipe.
- SUPPORTED_PLATFORMS – Describe platforms, especially leadership platforms.
- Where is the product pre-installed? Facility and vendor stacks, Linux distros, etc.
- Pointer to usage stats: NERSC makes theirs' available, ORNL generates them but keeps internal.
- Documentation: Could be incorporated into README file, if brief. Or README could point to DOCUMENTATION file, which would contain documentation, point to detailed documentation or both.
- Could have an FAQ: Primary text with some content not found in the product special files, e.g., information about how Spack supports modules generation.
- VERSION - Contains the current product version number and the corresponding hash in the repo.

Putting it all together: Products + SDKs + E4S

- In ECP, teams increasingly need to ensure that their libraries and components work together
 - Historically, HPC codes used very few dependencies
- Now, groups of teams work together on small releases of “Software Development Kits”
- SDKs will be rolled into a larger, periodic release.



Preparing for Exascale Platforms



Department of Energy (DOE) Roadmap to Exascale Systems

An impressive, productive lineup of *accelerated node* systems supporting DOE's mission

Pre-Exascale Systems [Aggregate Linpack (Rmax) = 323 PF!]

First U.S. Exascale Systems

2012

2016

2018

2020

2021-2023



Titan (9)

ORNL

Cray/AMD/NVIDIA



Mira (21)

ANL

IBM BG/Q



Theta (24)

ANL

Cray/Intel KNL



Cori (12)

LBNL

Cray/Intel Xeon/KNL



Summit (1)

ORNL

IBM/NVIDIA



Perlmutter

LBNL

Cray/AMD/NVIDIA



ORNL

TBD



Aurora

ANL

Intel/Cray



Sequoia (10)

LLNL

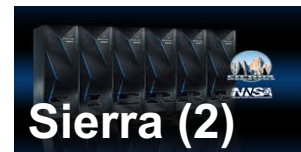
IBM BG/Q



Trinity (6)

LANL/SNL

Cray/Intel Xeon/KNL



Sierra (2)

LLNL

IBM/NVIDIA



CROSSROADS

LANL/SNL

TBD



EL CAPITAN

LLNL

TBD

Some Observations on Accelerator Programming

- Wait until you are ready.
 - A lot of flux in software stacks.
 - Let others (us) work on stabilizing the ecosystem.
- Start with a concrete, robust development environment.
 - Many people start with CUDA, even if the target is different.
 - Portability abstractions, e.g., Kokkos, can be reasonable too.
- Expose massive concurrency, hiding massive latency
 - GPUs are multi tera-operation per second devices.
 - Need a lot of uninterrupted work to hide overheads.
- Note: Designing for GPUs will also generally improve CPU performance.
 - Modern CPUs benefit a lot from concurrency.
 - CPU memory systems perform better with more concurrent requests.

Working out Our On-node Parallel Programming Strategies

- CUDA/HIP/SYCL/OpenCL – Explicit parallel threading.
 - We made the transition to using CUDA. Supporting AMD, Intel challenging.
- OpenMP – Markup-based approach
 - Need high tolerance for markup, performance portability uncertain.
- Kokkos – parallel_for/reduce/scan patterns
 - User provides loop body, C++ functor/lambda.
- Side note on C – has long been a parallel programming language.
 - Speculative execution, branch prediction
 - “C is not a low-level language”, David Chisnall.
<http://delivery.acm.org/10.1145/3220000/3212479/p10-chisnall.pdf>

Making Transition to Accelerators: xSDK ECP Math Libraries Porting Efforts

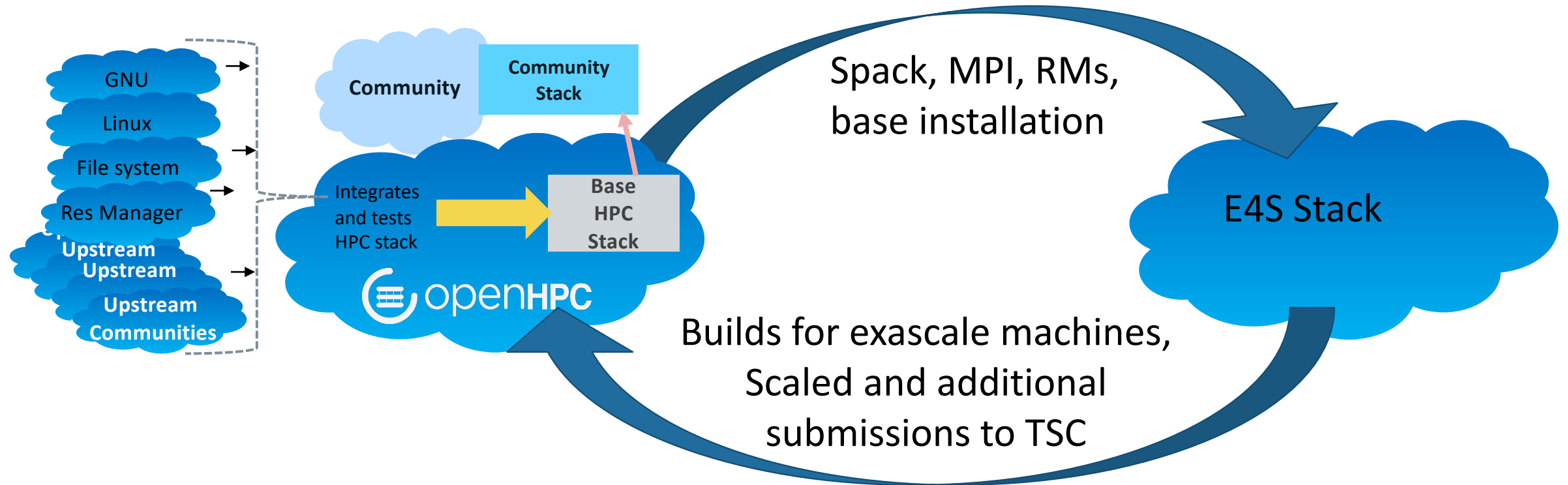


Package	NVIDIA GPU	AMD GPU	Intel Xe
DTK	support (Kokkos)	in progress (Kokkos)	in progress (Kokkos)
Ginkgo	support (CUDA)	in progress (HIP)	no support
hypre	support (CUDA/OMP4.5/RAJA/Kokkos)	no support	no support
MAGMA	support (CUDA)	support (OpenCL)	in progress (OpenCL/SYCL)
MFEM	support (CUDA/OCCA/RAJA)	support (HIP/OCCA)	in progress (OCCA/OpenCL)
PETSc	support (CUDA)	support (ViennaCL/OpenCL)	no support
STRUMPACK	in progress (CUDA)	no support	no support
SUNDIALS	support (CUDA/RAJA/OMP4.5)	no support	in progress (OMP4.5)
SuperLU	support (CUDA)	no support	no support
Tasmanian	support (CUDA)	no support	no support
Trilinos	support (Kokkos)	in progress (Kokkos)	in progress (Kokkos)

E4S Collaborations



How E4S and OpenHPC Work Together



• OpenHPC

- Provides installation and base components for installing and running cluster to supercomputer
- Provides generic binaries for all systems (working on targeted builds)

• E4S

- Built from scratch for your system
- Focus on upper layers of system software stack (libraries, runtimes, etc.)
- Targeted for capability-class machines

Dates: July 10-11, 2019

Location: 312 Lillis, University of Oregon, Eugene, OR 97403

Workshop Agenda

Wednesday, July 10, 2019: 312 Lillis, University of Oregon

8:00am - 8:45am: Registration

8:45am - 9:15am: Welcome, introductions, David Conover (VPRI, UO), Vipin Chaudhary (Program Director, CISE/OAC, NSF), Jonathan Carter (Deputy Director, Software Technology, ECP, and Deputy for Science, LBL) [\[slides\]](#) [\[video\]](#)

9:15am - 10:30am: Spack tutorial - Todd Gamblin and Greg Becker (LLNL) [\[slides\]](#) [\[video\]](#)

10:30am - 11:00am: Break

11:00am - 12:30pm: Spack tutorial - Todd Gamblin (LLNL) [\[video\]](#)

12:30pm - 2:00pm: Lunch break: 440 Lillis

2:00pm - 3:30pm: Spack tutorial - Todd Gamblin and Greg Becker (LLNL) [\[video\]](#)

3:30 pm - 4:00pm: Break

4:00pm - 5:30pm: Spack tutorial - Todd Gamblin and Greg Becker (LLNL) [\[video 1\]](#) [\[video 2\]](#) [\[video 3\]](#) [\[video 4\]](#) [\[video 5\]](#)

6:30pm - 9:30pm: Working dinner at the [Jordan Schnitzer Museum of Art, UO](#)

<https://oaciss.uoregon.edu/NSFDOE19/agenda.html>

Thursday, July 11, 2019: 312 Lillis, University of Oregon

9:00am - 9:30am: Unifying Software Distribution in ECP - Todd Gamblin (LLNL) [\[slides\]](#) [\[video\]](#)

9:00am - 9:30am: Containers for HPC - Andrew Younge (Sandia National Laboratories, NM) [\[slides\]](#) [\[video\]](#)

9:30am - 10:00am: Software deployment at DOE facilities - David Montoya (Los Alamos National Laboratory, NM) [\[slides\]](#) [\[video\]](#)

10:00am - 10:30am: E4S - Sameer Shende (University of Oregon) [\[slides\]](#) [\[video\]](#)

10:30am - 11:00am: Break

11:00am - 11:30am: Overview of software architecture - Todd Gamblin, LLNL

11:30am - 12:30pm: Hands-on, applying Spack to applications.

12:30pm - 2:00pm: Lunch: 440 Lillis

2:00pm - 3:30pm: Hands-on, applying Spack to applications.

3:30pm - 4:00pm: Break

4:00pm - 5:00pm: Hands-on, Spack and E4S.

5:00pm - 6:15pm: Closing remarks, planning for the next workshop - Jonathan Carter (Lawrence Berkeley National Laboratory)

6:30pm: Dinner at [Excelsior Inn](#)

Extending Collaborations

- E4S/SDK collaborations make sense with:
 - HPC open source development projects:
 - deal.II (NSF math library),
 - PHIST (DLR Germany library).
 - Application teams in search of open source software foundation:
 - NSF science teams.
 - NOAA, others.
 - Commercial open source packagers/distributors
 - OpenHPC.
 - HPC systems vendors.
 - HPC systems facilities:
 - SDSC, TACC, others.

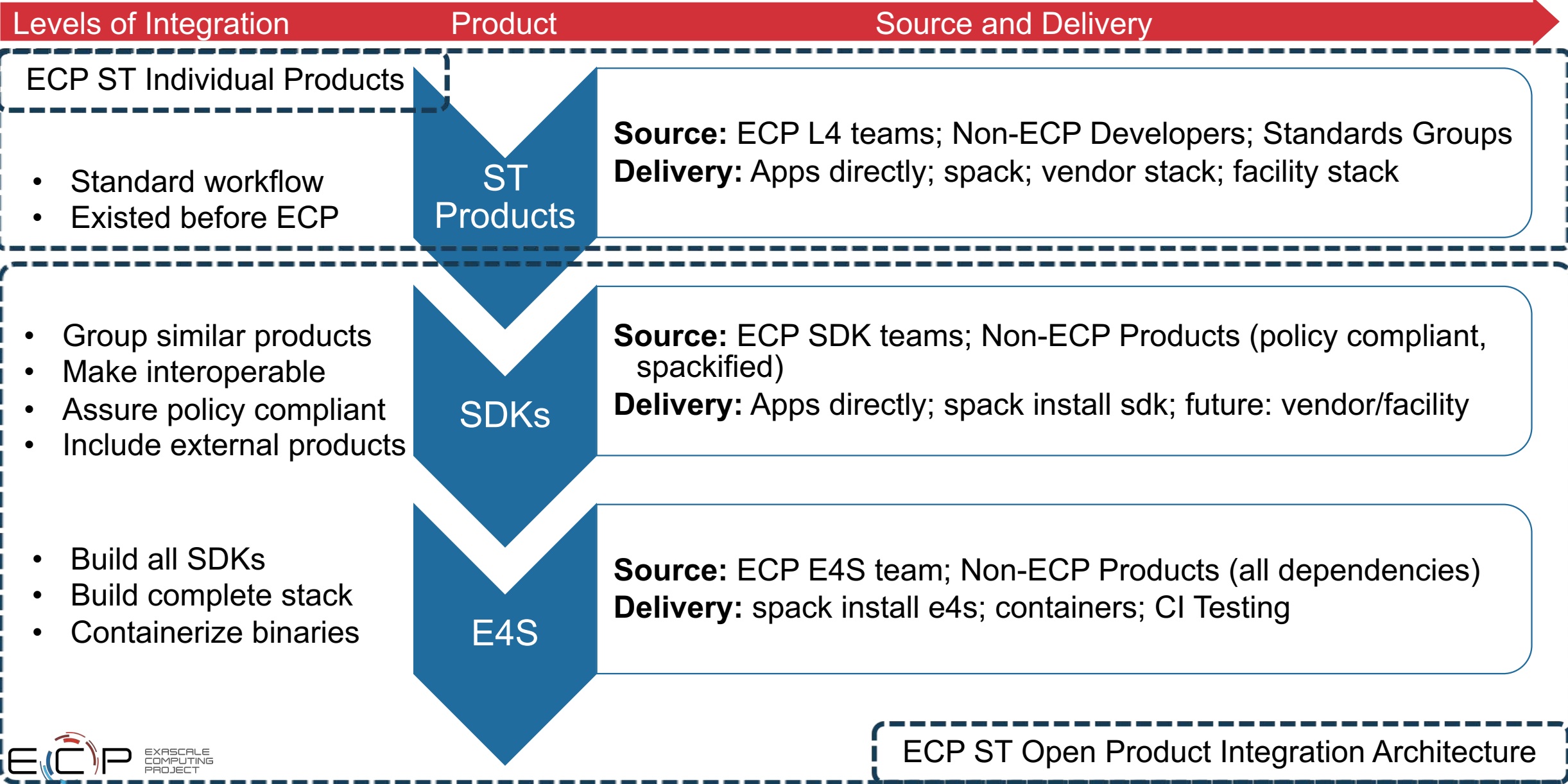
E4S: Building on top of previous efforts

- E4S did not emerge from nothing.
- Leveraging the work of many others.
- HPC Linux: Work done at U of Oregon, and at ParaTools.
- IDEAS-Classic: xSDK – the original SDK continuing under ECP.
- Spack – Pre-dates E4S.

E4S Wrap Up



Software Technology Ecosystem



E4S Summary

What E4S is not

- A closed system taking contributions only from DOE software development teams.
- A monolithic, take-it-or-leave-it software behemoth.
- A commercial product.
- A simple packaging of existing software.

What E4S is

- Extensible, open architecture software ecosystem accepting contributions from US and international teams.
- Framework for collaborative open-source product integration.
- A full collection of compatible software capabilities **and**
- A manifest of a la carte selectable software capabilities.
- Vehicle for delivering high-quality reusable software products in collaboration with others.
- The conduit for future leading edge HPC software targeting scalable next-generation computing platforms.
- A hierarchical software framework to enhance (via SDKs) software interoperability and quality expectations.

Final Remarks

- Software Development Kits (SDKs) provide a new software coordination mechanism:
 - Do not significantly increase the number of software products in the HPC ecosystem.
 - Provide intermediate build, install and test targets for E4S (reducing complexity).
 - Enable development teams to improve and standardize look-and-feel, best practices, policies.
 - Improve interoperability, interchangeability of similar products.
 - Fosters communities of developers in a cooperative/competitive environment.
 - **Provides integration point for SDK-compatible, non-ECP products.**
- The Extreme-scale Scientific Software Stack (E4S) provides a complete HPC software stack:
 - Does not significantly increase the number of software products in the HPC ecosystem.
 - Provides a coordinated approach to building, installing and testing HPC software.
 - Tests some products with a subset of configurations on a subset of platforms.
 - Improves stability of the ST stack so that any subset is more stable.
 - **Provides a complete software stack, including non-ECP products.**
- **The SDK/E4S architecture is open:**
 - Enables light-weight coordinated collaboration among open source software teams.
 - ECP seeks collaboration: libraries/tools and SDKs, facilities and E4S.